11-411
Natural Language Processing
Overview

Kemal Oflazer

Carnegie Mellon University in Qatar

[*] Content mostly based on previous offerings of 11-411 by LTI Faculty at CMU-Pittsburgh.

What is NLP?

- Automating the analysis, generation, and acquisition of human (”natural”) language
  - Analysis (or “understanding” or “processing”, . . .)
  - Generation
  - Acquisition
- Some people use “NLP” to mean all of language technologies.
- Some people use it only to refer to analysis.

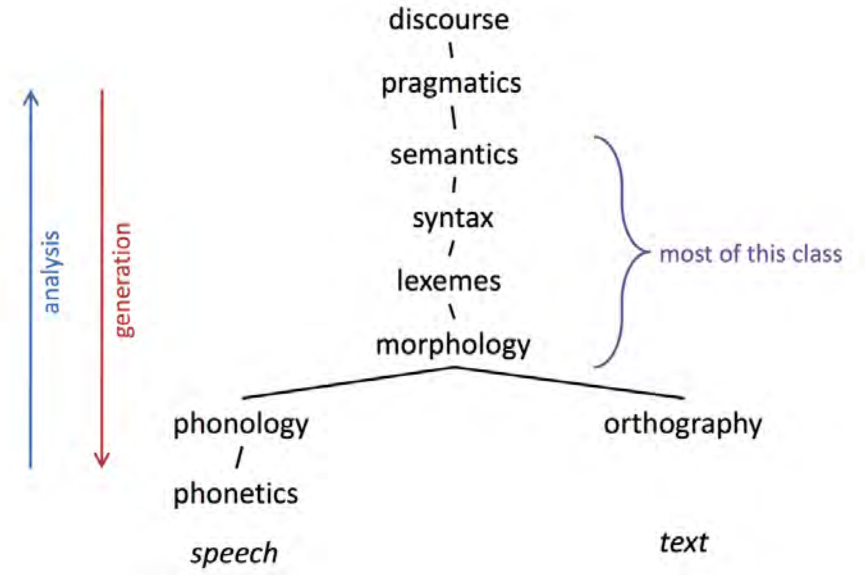
Why NLP?

- Answer questions using the Web
- Translate documents from one language to another
- Do library research; summarize
- Manage messages intelligently
- Help make informed decisions
- Follow directions given by any user
- Fix your spelling or grammar
- Grade exams
- Write poems or novels
- Listen and give advice
- Estimate public opinion
- Read everything and make predictions
- Interactively help people learn
- Help disabled people
- Help refugees/disaster victims
- Document or reinvigorate indigenous languages

What is NLP? More Detailed Answer

- Automating language analysis, generation, acquisition.
  - Analysis (or “understanding” or “processing” ...): input is language, output is some representation that supports useful action
  - Generation: input is that representation, output is language
  - Acquisition: obtaining the representation and necessary algorithms, from knowledge and data
- Representation?

Levels of Linguistic Representation



Why it’s Hard

- The mappings between levels are extremely complex.
- Details and appropriateness of a representation depends on the application.

Complexity of Linguistics Representations

- Input is likely to be noisy.
- Linguistic representations are theorized constructs; we cannot observe them directly.
- **Ambiguity**: each linguistic input may have many possible interpretations at every level.
  - The correct resolution of the ambiguity will depend on the intended meaning, which is often inferable from context.
  - People are good at linguistic ambiguity resolution.
  - Computers are not so good at it.
- How do we represent sets of possible alternatives?
- How do we represent context?

Complexity of Linguistics Representations

- **Richness**: there are many ways to express the same meaning, and immeasurably many meanings to express. Lots of words/phrases.
- Each level interacts with the others.
- There is tremendous diversity in human languages.
    - Languages express the same kind of meaning in different ways
    - Some languages express some meanings more readily/often.
- We will study models.

What is a Model?

- An abstract, theoretical, predictive construct. Includes:
  - a (partial) representation of the world
  - a method for creating or recognizing worlds,
  - a system for reasoning about worlds
- NLP uses many tools for modeling.
- Surprisingly, shallow models work fine for some applications.

Using NLP Models and Tools

- This course is meant to introduce some formal tools that will help you navigate the field of NLP.
- We focus on formalisms and algorithms.
  - This is not a comprehensive overview; it's a deep introduction to some key topics.
  - We'll focus mainly on analysis and mainly on English text (but will provide examples from other languages whenever meaningful)
  - The skills you develop will apply to any subfield of NLP

Applications / Challenges

- Application tasks evolve and are often hard to define formally.
- Objective evaluations of system performance are always up for debate.
  - This holds for NL analysis as well as application tasks.
- Different applications may require different kinds of representations at different levels.

Expectations from NLP Systems

- Sensitivity to a wide range of the phenomena and constraints in human language
- Generality across different languages, genres, styles, and modalities
- Computational efficiency at construction time and runtime
- Strong formal guarantees (e.g., convergence, statistical efficiency, consistency, etc.)
- High accuracy when judged against expert annotations and/or task-specific performance

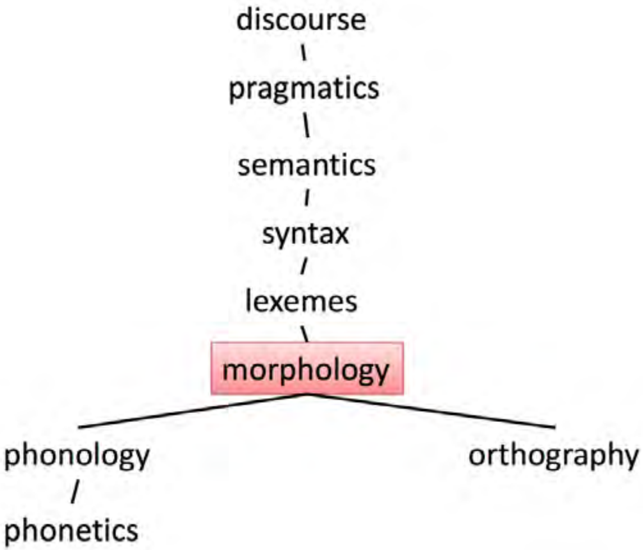
Key Applications (2017)

- Computational linguistics (i.e., modeling the human capacity for language computationally)
- Information extraction, especially “open” IE
- Question answering (e.g., Watson)
- Conversational Agents (e.g., Siri, OK Google)
- Machine translation
- Machine reading
- Summarization
- Opinion and sentiment analysis
- Social media analysis
- Fake news detection
- Essay evaluation
- Mining legal, medical, or scholarly literature

NLP vs Computational Linguistics

- NLP is focussed on the technology of processing language
- Computational Linguistics is focussed on using technology to support/implement linguistics.
- The distinction is
  - Like “artificial intelligence” vs. “cognitive science”
  - Like “ building airplanes” vs. “understanding how birds fly”

Let’s Look at Some of the Levels



discourse

pragmatics

semantics

syntax

lexemes

morphology

phonology

orthography

phonetics

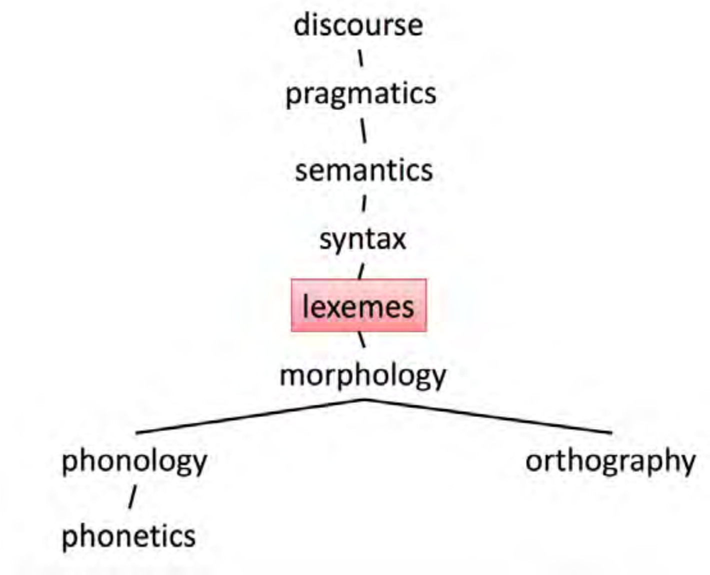
Morphology

- Analysis of words into meaningful components – morphemes.
- Spectrum of complexity across languages
- **Isolating Languages**: mostly one morpheme (e.g., Chinese/Mandarin)
- **Inflectional Languages**: mostly two morphemes (e.g., English, French, one morpheme may mean many things)
  - go+ing, habla+mos “I have spoken” (SP)

Morphology

- **Agglutinative Languages**: Mostly many morphemes stacked like "beads-on-a-string" (e.g., Turkish, Finnish, Hungarian, Swahili)
  - uygar+laş+tır+ama+dık+lar+ımız+dan+mış+sınız+casına
    "(behaving) as if you are among those whom we could not civilize"
- **Polysynthetic Languages**: A word is a sentence! (e.g., Inuktikut)
  - Parismunngaujumaniralauqsimanngittunga
    Paris+mut+nngau+juma+niraq+lauq+si+ma+nngit+jun
    "I never said that I wanted to go to Paris"
- Reasonably dynamic:
  - unfriend, Obamacare

Let’s Look at Some of the Levels



discourse

pragmatics

semantics

syntax

lexemes

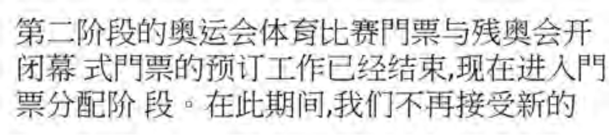
morphology

phonology

orthography

phonetics

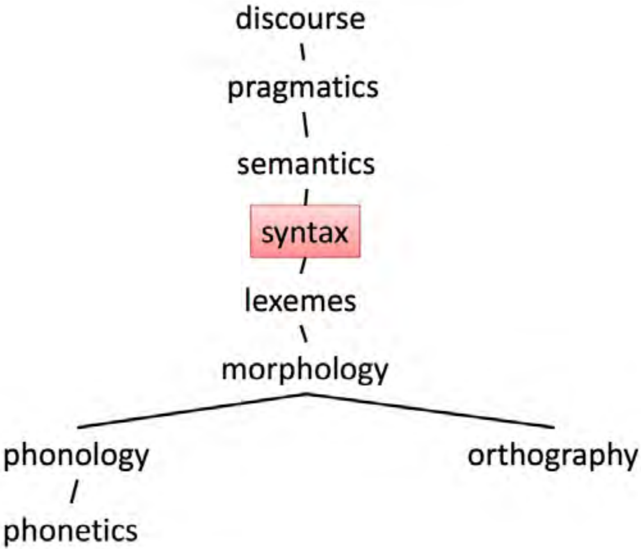
Lexical Processing



第二阶段的奥运会体育比赛門票与残奥会开
闭幕 式門票的预订工作已经结束,现在进入門
票分配阶 段。在此期间,我们不再接受新的

- Segmentation
- Normalize and disambiguate words
  - Words with multiple meanings: bank, mean
    - Extra challenge: domain-specific meanings (e.g., *latex*)
  - Process multi-word expressions
    - make . . . decision, take out, make up, kick the . . . bucket
- Part-of-speech tagging
  - Assign a syntactic class to each word (verb, noun, adjective, etc.)
- Supersense tagging
  - Assign a coarse semantic category to each content word (motion event, instrument, foodstuff, etc.)
- Syntactic “supertagging”
  - Assign a possible syntactic neighborhood tag to each word (e.g., subject of a verb)

Let’s Look at Some of the Levels



discourse

pragmatics

semantics

syntax

lexemes

morphology

phonology

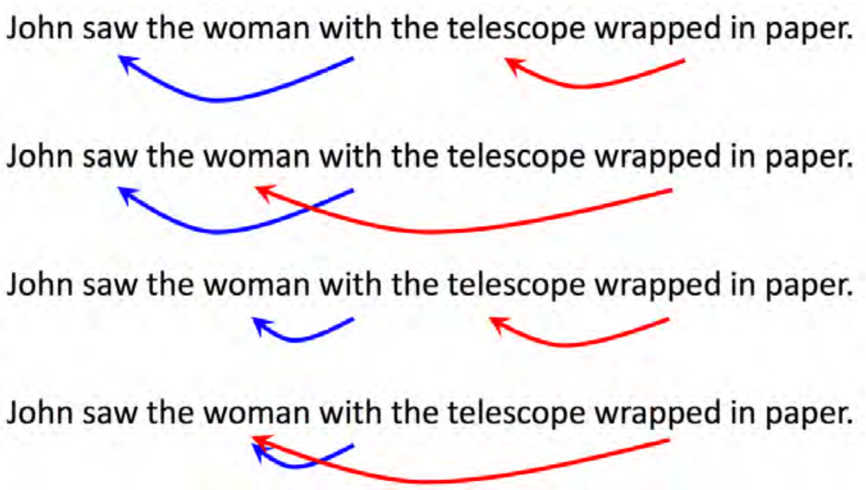
orthography

phonetics

Syntax

- Transform a sequence of symbols into a hierarchical or compositional structure.
- Some sequences are well-formed, others are not
  - ✓ I want a flight to Tokyo.
  - ✓ I want to fly to Tokyo.
  - ✓ I found a flight to Tokyo.
  - × I found to fly to Tokyo.
  - ✓ Colorless green ideas sleep furiously.
  - × Sleep colorless green furiously ideas.
- Ambiguities explode combinatorially
  - Students hate annoying professors.
  - John saw the woman with the telescope.
  - John saw the woman with the telescope wrapped in paper.

Some of the Possible Syntactic Analyses



John saw the woman with the telescope wrapped in paper.

John saw the woman with the telescope wrapped in paper.

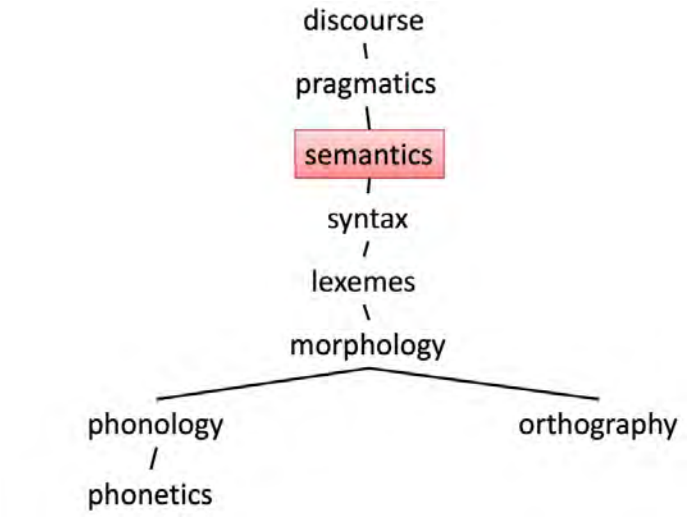
John saw the woman with the telescope wrapped in paper.

John saw the woman with the telescope wrapped in paper.

Morphology–Syntax

- A ship-shipping ship, shipping shipping-ships.

Let’s Look at Some of the Levels



discourse

pragmatics

semantics

syntax

lexemes

morphology

phonology

orthography

phonetics

Semantics

- Mapping of natural language sentences into domain representations.
  - For example, a robot command language, a database query, or an expression in a formal logic
- Scope ambiguities:
  - In this country a woman gives birth every fifteen minutes.
  - Every person on this island speaks three languages.
  - (TR) Üç doktor her hastaya baktı “Three doctors every patient saw”

$$\Rightarrow \exists d_1, d_2, d_3, doctor(d_1)\&doctor(d_1)\&doctor(d_1)\ (\forall p, patient(p), saw(d_1,p)\&saw(d_2,p)\&saw(d_3,p))$$

  - (TR) Her hastaya üç doktor baktı “Every patient three doctors saw”

$$\Rightarrow \forall p, patient(p)(\exists d_1, d_2, d_3, doctor(d_1)\&doctor(d_1)\&doctor(d_1)\&saw(d_1,p)\&saw(d_2,p)\&saw(d_3,p))$$

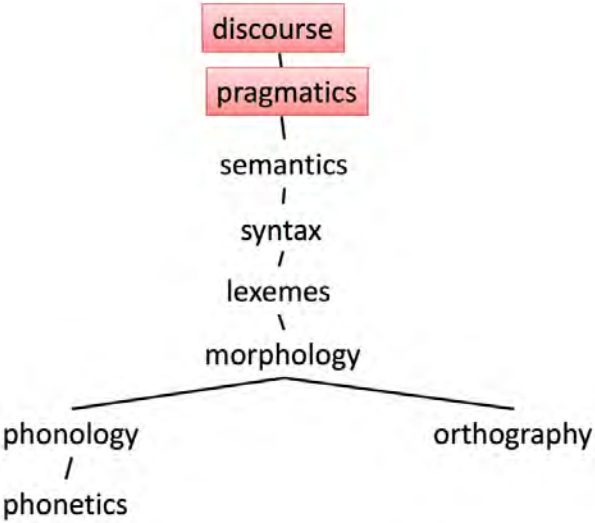
- Going beyond specific domains is a goal of general artificial Intelligence.

Syntax–Semantics

We saw the woman with the telescope wrapped in paper.

- Who has the telescope?
- Who or what is wrapped in paper?
- Is this an event of perception or an assault?

Let’s Look at Some of the Levels



Pragmatics/Discourse

- Pragmatics
  - Any non-local meaning phenomena
    - “Can you pass the salt?”
    - “Is he 21?” “Yes, he’s 25.”
- Discourse
  - Structures and effects in related sequences of sentences
    - “I said the **black** shoes.”
    - “Oh, **black**.” (Is that a sentence?)

Course Logistics/Administrivia

- Web page: `piazza.com/qatar.cmu/fall2017/11411/home`
- Course Material:
  - Book: Speech and Language Processing, Jurafsky and Martin, 2nd ed.
  - As needed, copies of papers, etc. will be provided.
  - Lectures slides will be provided after each lecture.
- Instructor: Kemal Oflazer

Your Grade

- Class project, 30%
- In-class midterm (October 11, for the time being), 20%
- Final exam (date TBD), 20%
- Unpredictable in-class quizzes, 15%
- Homework assignments, 15%

Policies

- Everything you submit must be your own work
- Any outside resources (books, research papers, web sites, etc.) or collaboration (students, professors, etc.) must be explicitly acknowledged.
- Project
  - Collaboration is required (team size TBD)
  - It's okay to use existing tools, but you must acknowledge them.
  - Grade is mostly shared.
  - Programming language is up to you.
- Do people know Python? Perl?

11-411
Natural Language Processing
Applications of NLP

Kemal Oflazer

Carnegie Mellon University in Qatar

* Content mostly based on previous offerings of 11-411 by LTI Faculty at CMU-Pittsburgh.

Information Extraction – Bird's Eye View

- **Input:** text, empty relational database
- **Output:** populated relational database

Senator John Edwards is to drop out of the race to become the Democratic party's presidential candidate after consistently trailing in third place. In the latest primary, held in Florida yesterday, Edwards gained only 14% of the vote, with Hillary Clinton polling 50% and Barack Obama on 33%. A reported 1.5m voters turned out to vote.

⇒

State	**Party**	**Cand.**	**%**
FL	Dem.	Edwards	14
FL	Dem.	Clinton	50
FL	Dem.	Obama	33

Named-Entity Recognition

- **Input:** text
- **Output:** text annotated with named-entities

Senator John Edwards is to drop out of the race to become the Democratic party's presidential candidate after consistently trailing in third place. In the latest primary, held in Florida yesterday, Edwards gained only 14% of the vote, with Hillary Clinton polling 50% and Barack Obama on 33%. A reported 1.5m voters turned out to vote.

$\Longrightarrow$

$[_{PER}$ **Senator John Edwards**] is to drop out of the race to become the $[_{GPE}$ **Democratic party**]'s presidential candidate after consistently trailing in third place. In the latest primary, held in $[_{LOC}$ **Florida**] yesterday, $[_{PER}$ **Edwards**] gained only 14% of the vote, with $[_{PER}$ **Hillary Clinton**] polling 50% and $[_{PER}$ **Barack Obama**] on 33%. A reported 1.5m voters turned out to vote.

Reference Resolution

- **Input:** text possibly with annotated named-entities
- **Output:** text annotated with named-entities and the real-world entitities they refer to.

[$_{PER}$ **Senator John Edwards**] is to drop out of the race to become the [$_{GPE}$ **Democratic party**]'s presidential candidate after consistently trailing in third place. In the latest primary, held in [$_{LOC}$ **Florida**] yesterday, [$_{PER}$ **Edwards**] gained only 14% of the vote, with [$_{PER}$ **Hillary Clinton**] polling 50% and [$_{PER}$ **Barack Obama**] on 33%. A reported 1.5m voters turned out to vote.

$\Rightarrow$

[$_{PER}$ **Senator John Edwards**] refers to

[$_{PER}$ **Edwards**] refers to

Coreference Resolution

- **Input:** text possibly with annotated named-entities
- **Output:** text with annotations of coreference chains.

[$_{PER}$ **Senator John Edwards**] is to drop out of the race to become the [$_{GPE}$ **Democratic party**]'s presidential candidate after consistently trailing in third place. In the latest primary, held in [$_{LOC}$ **Florida**] yesterday, [$_{PER}$ **Edwards**] gained only 14% of the vote, with [$_{PER}$ **Hillary Clinton**] polling 50% and [$_{PER}$ **Barack Obama**] on 33%. A reported 1.5m voters turned out to ote. This was a huge setback for the Senator from [$_{LOC}$ **North Carolina**].

$\Rightarrow$

[$_{PER}$ **Senator John Edwards**],
[$_{PER}$ **Edwards**]
Senator from [$_{LOC}$ **North Carolina**]
refer to the same entity.

Relation Extraction

- **Input:** text annotated with named-entitites
- **Output:** populated relational database with relations between entities.

Senator John Edwards is to drop out of the race to become the Democratic party's presidential candidate after consistently trailing in third place. In the latest primary, held in Florida yesterday, Edwards gained only 14% of the vote, with Hillary Clinton polling 50% and Barack Obama on 33%. A reported 1.5m voters turned out to vote.

⇒

Person	**Member-of**
John Edwards	Democrat Party
Hillary Clinton	Democrat Party
Barack Obama	Democrat Party

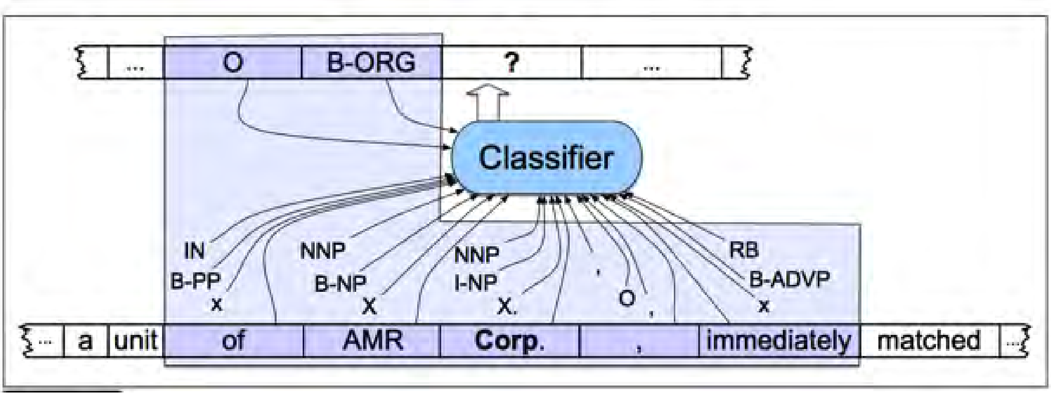
Encoding for Named-Entity Recognition

- Named-entity recognition is typically formulated as a *sequence tagging problem*.
- We somehow encode the *boundaries* and *types* of the named-entities.
- **BIO** Encoding
  - **B-*type*** indicates the beginning token/word of a named-entity (of type *type*)
  - **I-*type*** indicates (any) other tokens of a named-entity (length > 1)
  - **O** indicates that a token is *not* a part of any named-entity.
- **BIOLU** Encoding
  - **BIO** same as above
  - **L-*type*** indicates last token of a named-entity (length > 1)
  - **U-*type*** indicates a single token named-entity (length $= 1$)

Encoding for Named-Entity Recognition

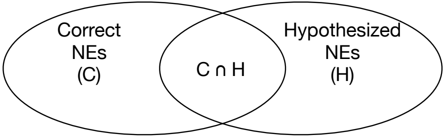
With	that	,	Edwards	’	campaign	will	end	the	way
O	O	O	B-PER	O	O	O	O	O	O
it	began	13	months	ago	–	with	the	candidate	pitching
O	O	O	O	O	O	O	O	O	O
in	to	rebuild	lives	in	a	city	still	ravaged	by
O	O	O	O	O	O	O	O	O	O
Hurricane	Katrina	.	Edwards	embraced	New	Orleans	as	a	glaring
B-NAT	I-NAT	O	B-PER	O	B-LOC	I-LOC	O	O	O
symbol	of	what	he	described	as	a	Washington	that	did
O	O	O	O	O	O	O	B-GPE	O	O
n’t	hear	the	cries	of	the	downtrodden	.		
O	O	O	O	O	O	O	O		

NER as a Sequence Modeling Problem



Evaluation of NER Performance

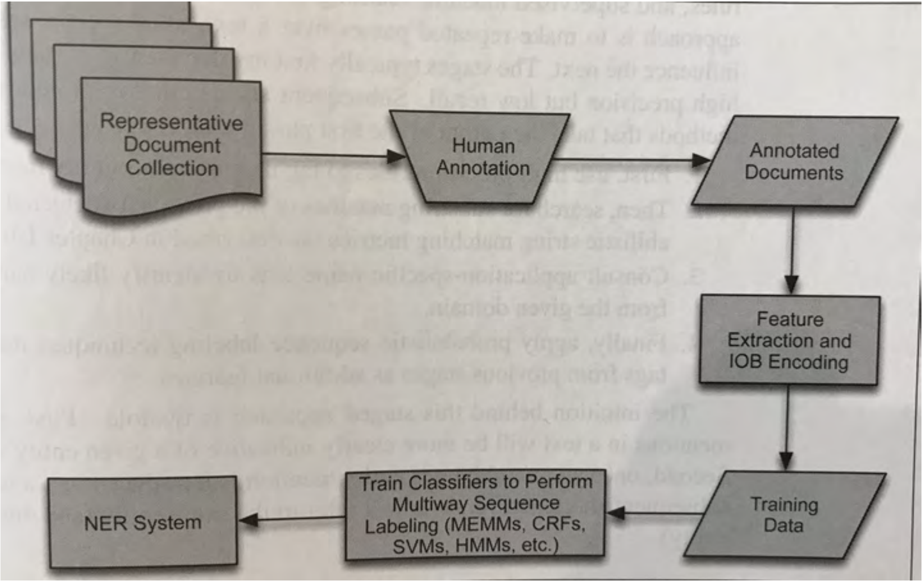
- Recall: What percentage of the actual named-entities did you correctly label?
- Precision: What percentage of the named-entities you labeled were actually correctly labeled?



$$R = \frac{|C \cap H|}{|C|} \qquad P = \frac{|C \cap H|}{|H|} \qquad F_1 = \frac{2 \cdot R \cdot P}{R + P}$$

- Actual: `[Microsoft Corp.]  CEO [Steve Ballmer] announced the release of [Windows 7] today`
- Tagged: `[Microsoft Corp.]  [CEO] [Steve] Ballmer announced the release of Windows 7 [today]`
- What is R, P, and F?

NER System Architecture


Representative Document Collection
Human Annotation
Annotated Documents
Feature Extraction and IOB Encoding
Training Data
Train Classifiers to Perform Multiway Sequence Labeling (MEMMs, CRFs, SVMs, HMMs, etc.)
NER System

Relation Extraction

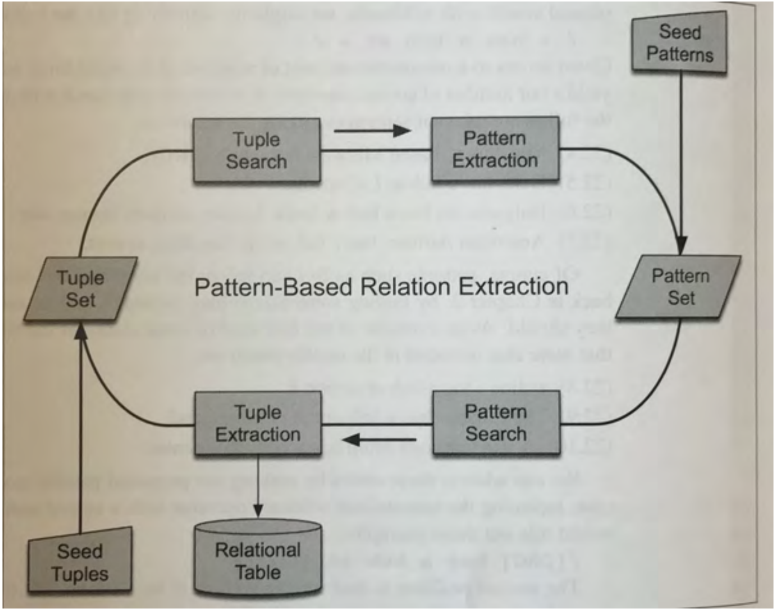
Some types of relations:

Relations		Examples	Type
Affiliations			
	Personal	*married to*, *mother of*	$PER \rightarrow PER$
	Organizational	*spokeman for*, *president of*	$PER \rightarrow ORG$
	Artifactual	*owns*, *invented*, *produces*	$(PER \mid ORG) \rightarrow ART$
Geospatial			
	Proximity	*near*, *on outskirts of*	$LOC \rightarrow LOC$
	Directional	*southeast of*	$LOC \rightarrow LOC$
Part-Of			
	Organizational	*unit of*, *parent-of*	$ORG \rightarrow ORG$
	Political	*annexed*, *acquired*	$GPE \rightarrow GPE$

Seeding Tuples

- Provide some examples
  - Brad is married to Angelina.
  - Bill is married to Hillary.
  - Hillary is married to Bill.
  - Hillary is the wife of Bill.
- Induce/provide seed patterns.
  - X is married to Y
  - X is the wife or Y
- Find other examples of X and Y mentioned closely and generate new patterns
  - Hillary and Bill wed in 1975 ⇒ X and Y wed

Bootstrapping Relations



Information Retrieval – the Vector Space Model

- Each document D_i is represented by a $|V|$-dimensional vector $\vec{d_i}$ (V is the vocabulary of words/tokens.)

$$\vec{d_i}[j] = \text{count of word } \omega_j \text{ in document } D_i$$

- A query Q is represented the same way with the vector $\vec{q}$:

$$\vec{q}[j] = \text{count of word } \omega_j \text{ in query } Q$$

- Vector Similarity $\Rightarrow$ Relevance of Document D_i to Query Q

$$\text{cosine\_similarity}(\vec{d_i}, \vec{q}) = \frac{\vec{d_i} \cdot \vec{q}}{\|\vec{d_i}\| \times \|\vec{q}\|}$$

- Twists: $tf - idf$ term frequency – inverse document frequency

$$x[j] = \text{count}(\omega_j) \times \log \frac{\#\text{ docs}}{\#\text{ docs with } \omega_j}$$

- Recall, Precision, Ranking

Information Retrieval – Evaluation

- Recall?

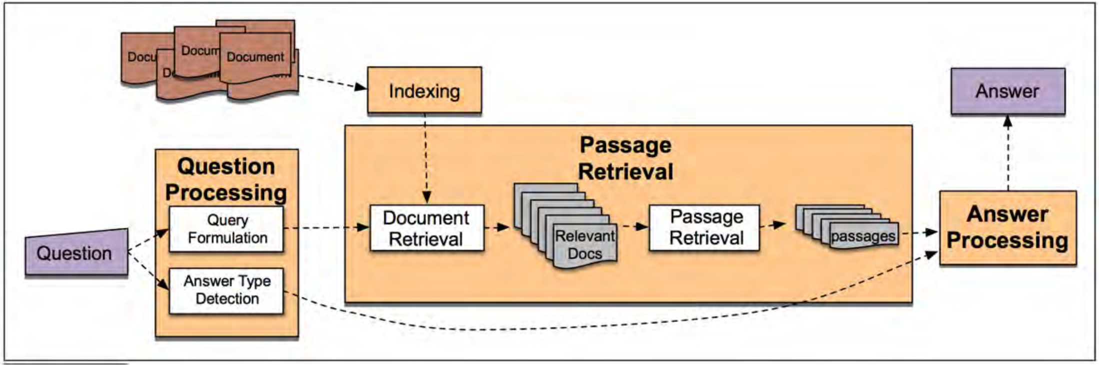
$$Recall = \frac{\text{Number of Relevant Documents Retrieved}}{\text{Number of Actual Relevant Documents in the Database}}$$

- Precision?

$$Precision = \frac{\text{Number of Relevant Documents Retrieved}}{\text{Number of Documents Retrieved}}$$

- Can you fool these?
- Are these useful? (Why did Google win the IR wars?)
- Ranking?
  - Is the “best” document close to the top if the list?

Question Answering



Question Answering Evaluation

- We typically get a list of answers back.
- Higher ranked correct answers are more valued.
- *Mean reciprocal rank*

$$\text{mean reciprocal rank} = \frac{1}{T}\sum_{i=1}^{T}\frac{1}{\text{rank of the first correct answer to question } i}$$

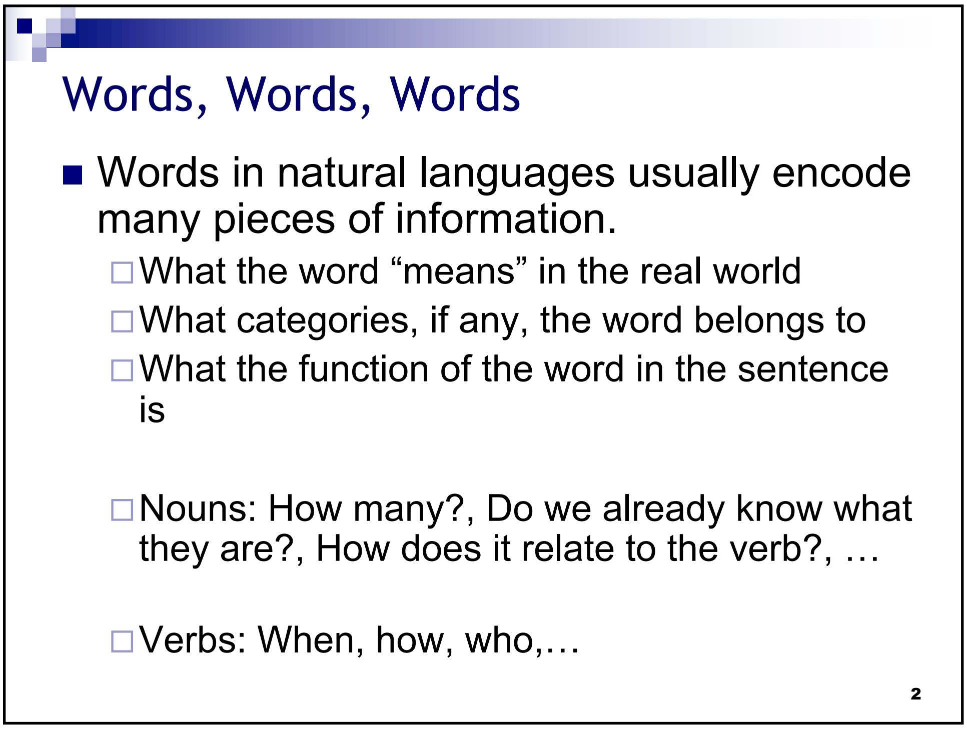
Some General Tools

- Supervised classification
- Feature vector representations
- Bootstrapping
- Evaluation:
  - Precision and recall (and their curves)
  - Mean reciprocal rank

11-411
Natural Language Processing

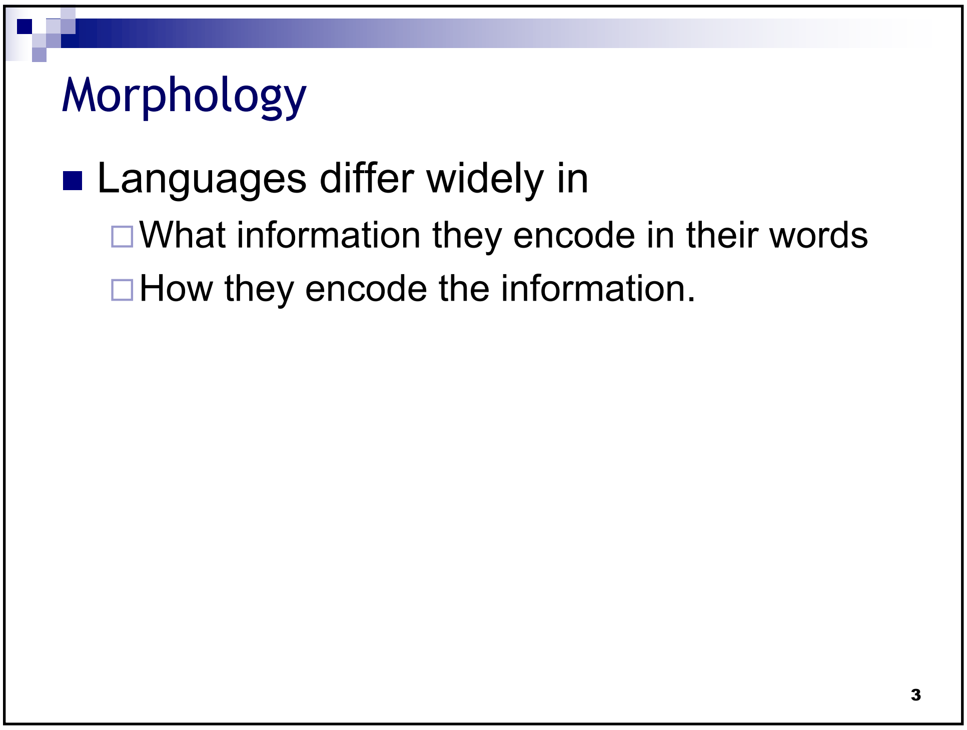
Words and Computational Morphology

Kemal Oflazer
Carnegie Mellon University - Qatar



Words, Words, Words

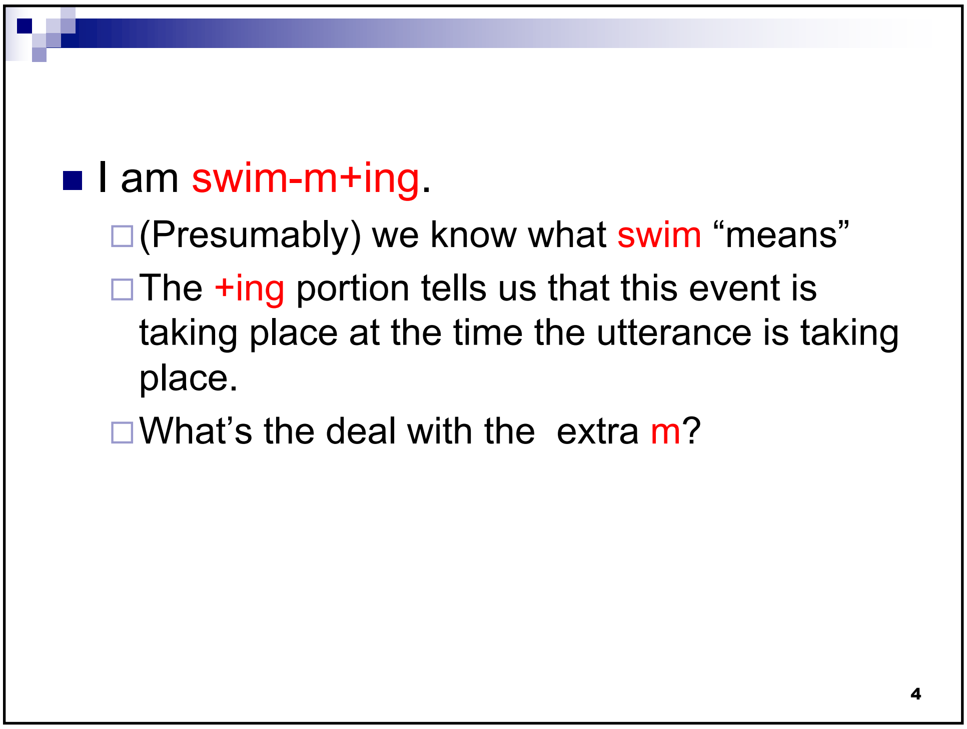
- Words in natural languages usually encode many pieces of information.
  - □ What the word “means” in the real world
  - □ What categories, if any, the word belongs to
  - □ What the function of the word in the sentence is
  - □ Nouns: How many?, Do we already know what they are?, How does it relate to the verb?, …
  - □ Verbs: When, how, who,…



Morphology

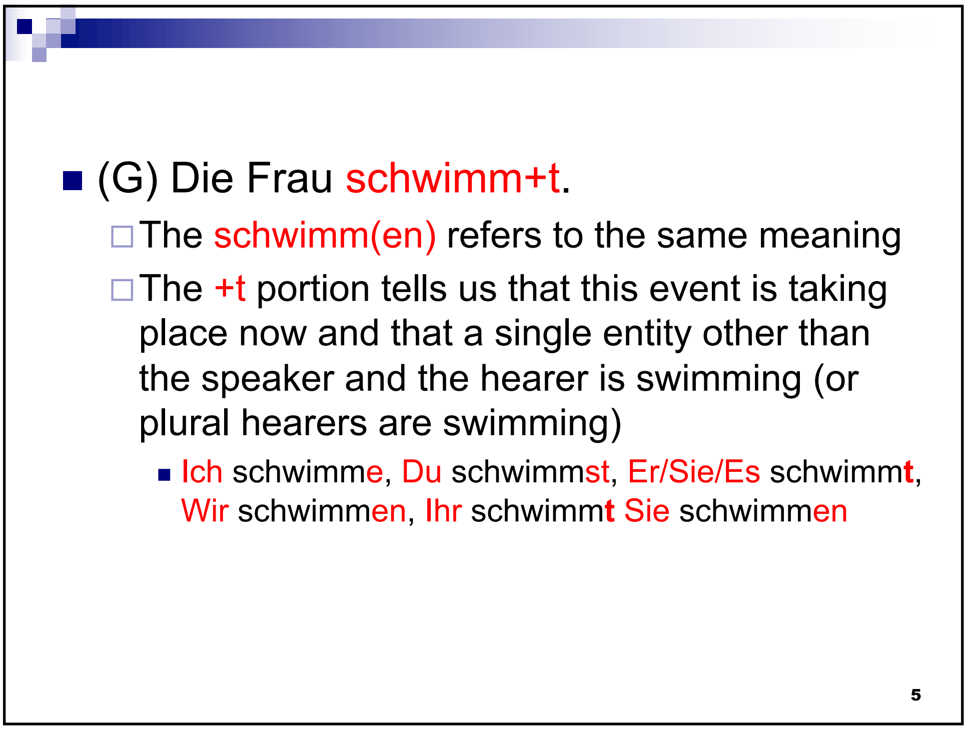
- Languages differ widely in
  - □ What information they encode in their words
  - □ How they encode the information.

3



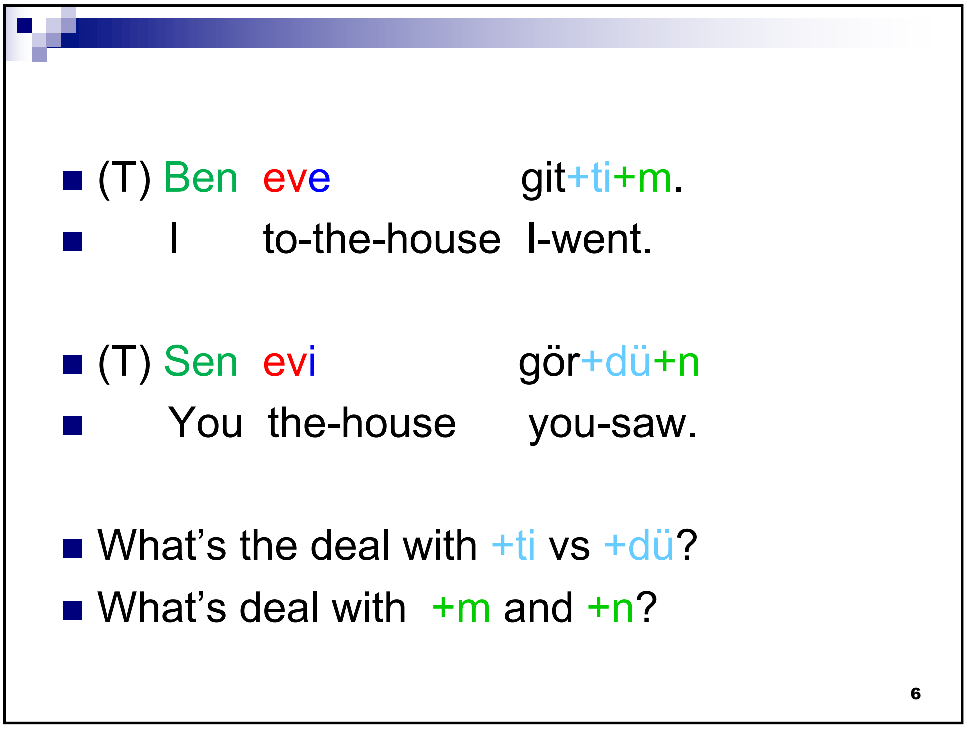
- I am swim-m+ing.
  - □ (Presumably) we know what swim "means"
  - □ The +ing portion tells us that this event is taking place at the time the utterance is taking place.
  - □ What's the deal with the extra m?

4



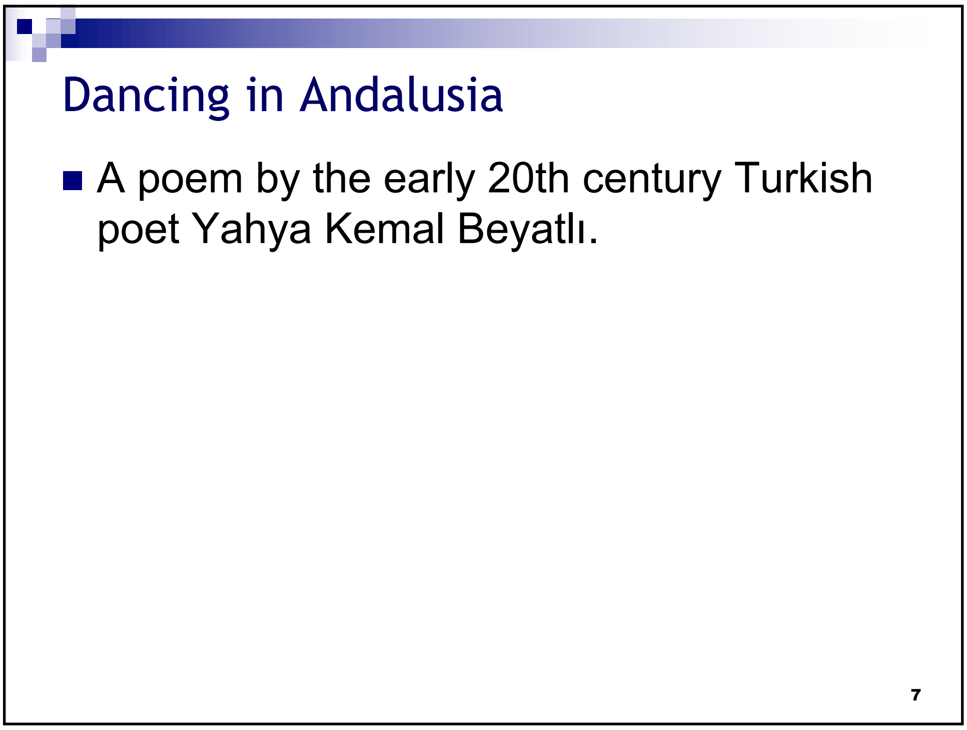
- (G) Die Frau schwimm+t.
  - □ The schwimm(en) refers to the same meaning
  - □ The +t portion tells us that this event is taking place now and that a single entity other than the speaker and the hearer is swimming (or plural hearers are swimming)
    - Ich schwimme, Du schwimmst, Er/Sie/Es schwimm**t**, Wir schwimmen, Ihr schwimm**t** Sie schwimmen

5



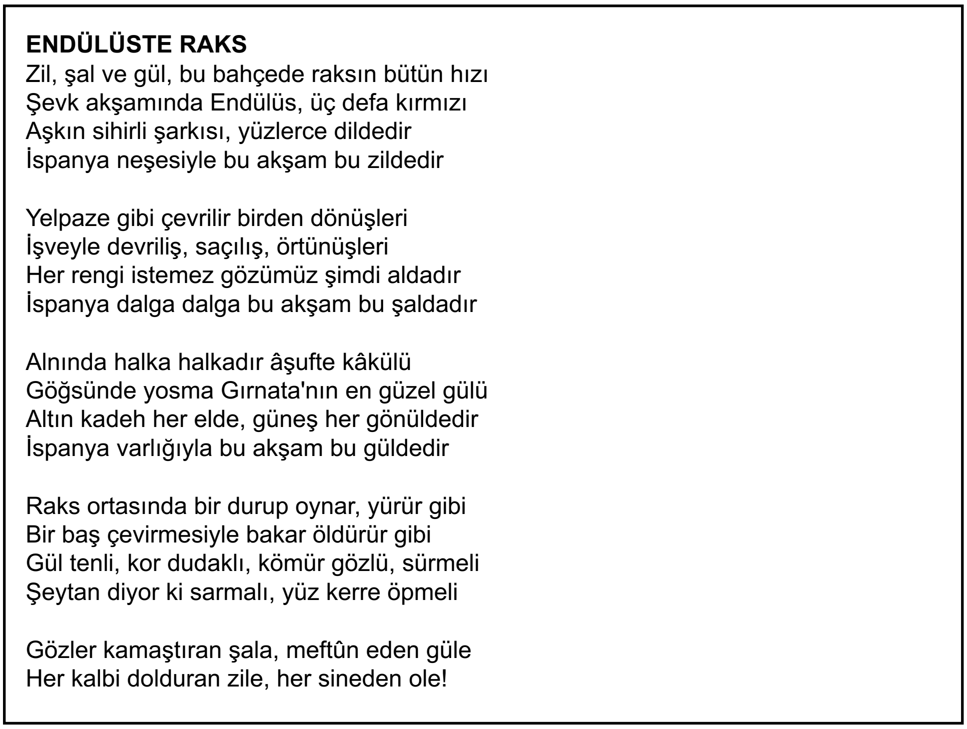
- (T) Ben eve git+ti+m.
- I to-the-house I-went.
- (T) Sen evi gör+dü+n
- You the-house you-saw.
- What’s the deal with +ti vs +dü?
- What’s deal with +m and +n?

6



Dancing in Andalusia

- A poem by the early 20th century Turkish poet Yahya Kemal Beyatlı.



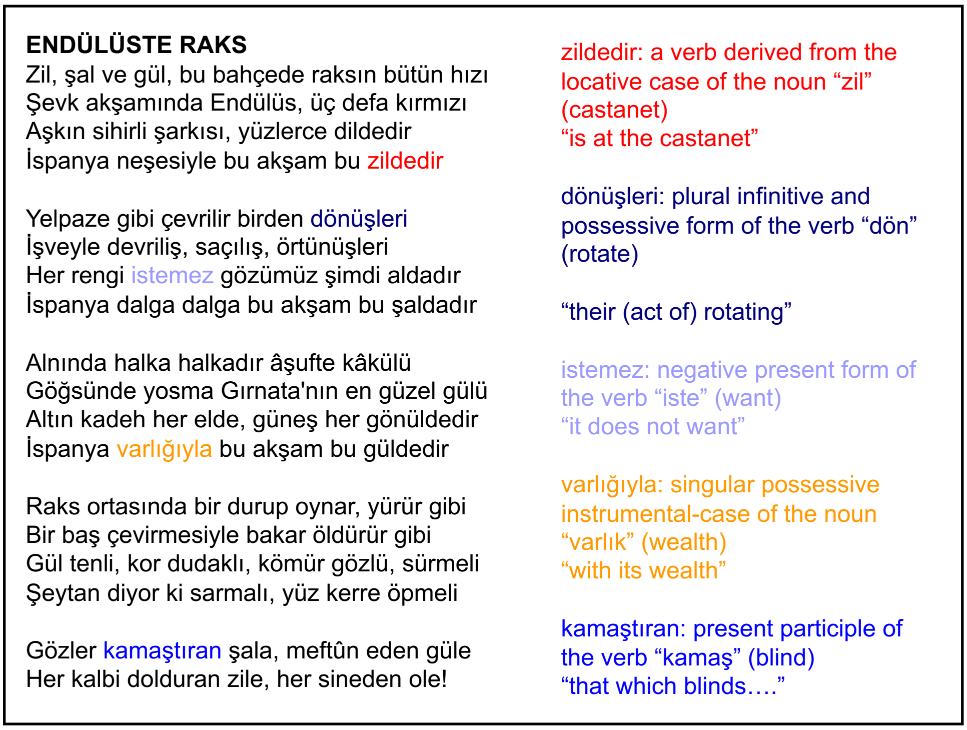
ENDÜLÜSTE RAKS
Zil, şal ve gül, bu bahçede raksın bütün hızı
Şevk akşamında Endülüs, üç defa kırmızı
Aşkın sihirli şarkısı, yüzlerce dildedir
İspanya neşesiyle bu akşam bu zildedir

Yelpaze gibi çevrilir birden dönüşleri
İşveyle devriliş, saçılış, örtünüşleri
Her rengi istemez gözümüz şimdi aldadır
İspanya dalga dalga bu akşam bu şaldadır

Alnında halka halkadır âşufte kâkülü
Göğsünde yosma Gırnata'nın en güzel gülü
Altın kadeh her elde, güneş her gönüldedir
İspanya varlığıyla bu akşam bu güldedir

Raks ortasında bir durup oynar, yürür gibi
Bir baş çevirmesiyle bakar öldürür gibi
Gül tenli, kor dudaklı, kömür gözlü, sürmeli
Şeytan diyor ki sarmalı, yüz kerre öpmeli

Gözler kamaştıran şala, meftûn eden güle
Her kalbi dolduran zile, her sineden ole!



ENDÜLÜSTE RAKS

Zil, şal ve gül, bu bahçede raksın bütün hızı
Şevk akşamında Endülüs, üç defa kırmızı
Aşkın sihirli şarkısı, yüzlerce dildedir
İspanya neşesiyle bu akşam bu zildedir

Yelpaze gibi çevrilir birden dönüşleri
İşveyle devriliş, saçılış, örtünüşleri
Her rengi istemez gözümüz şimdi aldadır
İspanya dalga dalga bu akşam bu şaldadır

Alnında halka halkadır âşufte kâkülü
Göğsünde yosma Gırnata'nın en güzel gülü
Altın kadeh her elde, güneş her gönüldedir
İspanya varlığıyla bu akşam bu güldedir

Raks ortasında bir durup oynar, yürür gibi
Bir baş çevirmesiyle bakar öldürür gibi
Gül tenli, kor dudaklı, kömür gözlü, sürmeli
Şeytan diyor ki sarmalı, yüz kerre öpmeli

Gözler kamaştıran şala, meftûn eden güle
Her kalbi dolduran zile, her sineden ole!

zildedir: a verb derived from the locative case of the noun “zil” (castanet)
“is at the castanet”

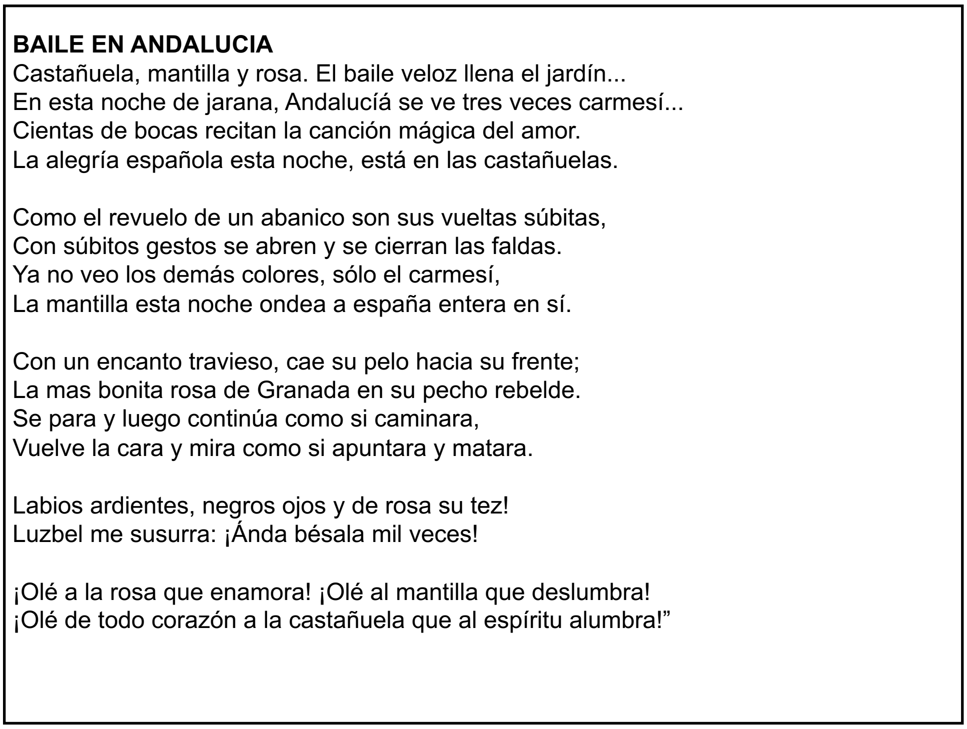
dönüşleri: plural infinitive and possessive form of the verb “dön” (rotate)

“their (act of) rotating”

istemez: negative present form of the verb “iste” (want)
“it does not want”

varlığıyla: singular possessive instrumental-case of the noun “varlık” (wealth)
“with its wealth”

kamaştıran: present participle of the verb “kamaş” (blind)
“that which blinds….”



BAILE EN ANDALUCIA

Castañuela, mantilla y rosa. El baile veloz llena el jardín...
En esta noche de jarana, Andalucíá se ve tres veces carmesí...
Cientas de bocas recitan la canción mágica del amor.
La alegría española esta noche, está en las castañuelas.

Como el revuelo de un abanico son sus vueltas súbitas,
Con súbitos gestos se abren y se cierran las faldas.
Ya no veo los demás colores, sólo el carmesí,
La mantilla esta noche ondea a españa entera en sí.

Con un encanto travieso, cae su pelo hacia su frente;
La mas bonita rosa de Granada en su pecho rebelde.
Se para y luego continúa como si caminara,
Vuelve la cara y mira como si apuntara y matara.

Labios ardientes, negros ojos y de rosa su tez!
Luzbel me susurra: ¡Ánda bésala mil veces!

¡Olé a la rosa que enamora! ¡Olé al mantilla que deslumbra!
¡Olé de todo corazón a la castañuela que al espíritu alumbra!”

BAILE EN ANDALUCIA
Castañuela, mantilla y rosa. El baile veloz llena el jardín...
En esta noche de jarana, Andalucíá se ve tres veces carmesí...
Cientas de bocas recitan la canción mágica del amor.
La alegría española esta noche, está en las castañuelas.

Como el revuelo de un abanico son sus vueltas súbitas,
Con súbitos gestos se abren y se cierran las faldas.
Ya no veo los demás colores, sólo el carmesí,
La mantilla esta noche ondea a españa entera en sí.

Con un encanto travieso, cae su pelo hacia su frente;
La mas bonita rosa de Granada en su pecho rebelde.
Se para y luego continúa como si caminara,
Vuelve la cara y mira como si apuntara y matara.

Labios ardientes, negros ojos y de rosa su tez!
Luzbel me susurra: ¡Ánda bésala mil veces!

¡Olé a la rosa que enamora! ¡Olé al mantilla que deslumbra!
¡Olé de todo corazón a la castañuela que al espíritu alumbra!"

castañuelas:
either
the plural feminine form of
the adjective "castañuelo"
(Castilian)
or
the plural of the feminine noun
"castañuela" (castanet)

BAILE EN ANDALUCIA
Castañuela, mantilla y rosa. El baile veloz llena el jardín...
En esta noche de jarana, Andalucíá se ve tres veces carmesí...
Cientas de bocas recitan la canción mágica del amor.
La alegría española esta noche, está en las castañuelas.

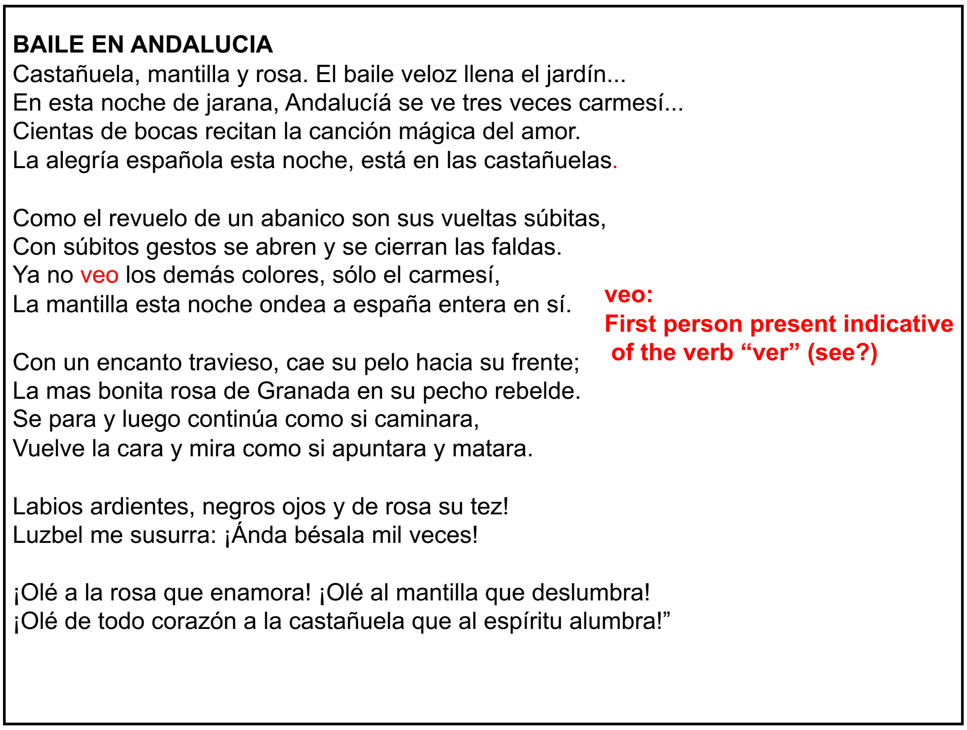
Como el revuelo de un abanico son sus vueltas súbitas,
Con súbitos gestos se abren y se cierran las faldas.
Ya no veo los demás colores, sólo el carmesí,
La mantilla esta noche ondea a españa entera en sí.

Con un encanto travieso, cae su pelo hacia su frente;
La mas bonita rosa de Granada en su pecho rebelde.
Se para y luego continúa como si caminara,
Vuelve la cara y mira como si apuntara y matara.

Labios ardientes, negros ojos y de rosa su tez!
Luzbel me susurra: ¡Ánda bésala mil veces!

¡Olé a la rosa que enamora! ¡Olé al mantilla que deslumbra!
¡Olé de todo corazón a la castañuela que al espíritu alumbra!"

vueltas:
either
the plural form of the
feminine noun "vuelta" (spin?)
or
the feminine plural
past participle of the verb
"volver"



BAILE EN ANDALUCIA
Castañuela, mantilla y rosa. El baile veloz llena el jardín...
En esta noche de jarana, Andalucíá se ve tres veces carmesí...
Cientas de bocas recitan la canción mágica del amor.
La alegría española esta noche, está en las castañuelas.

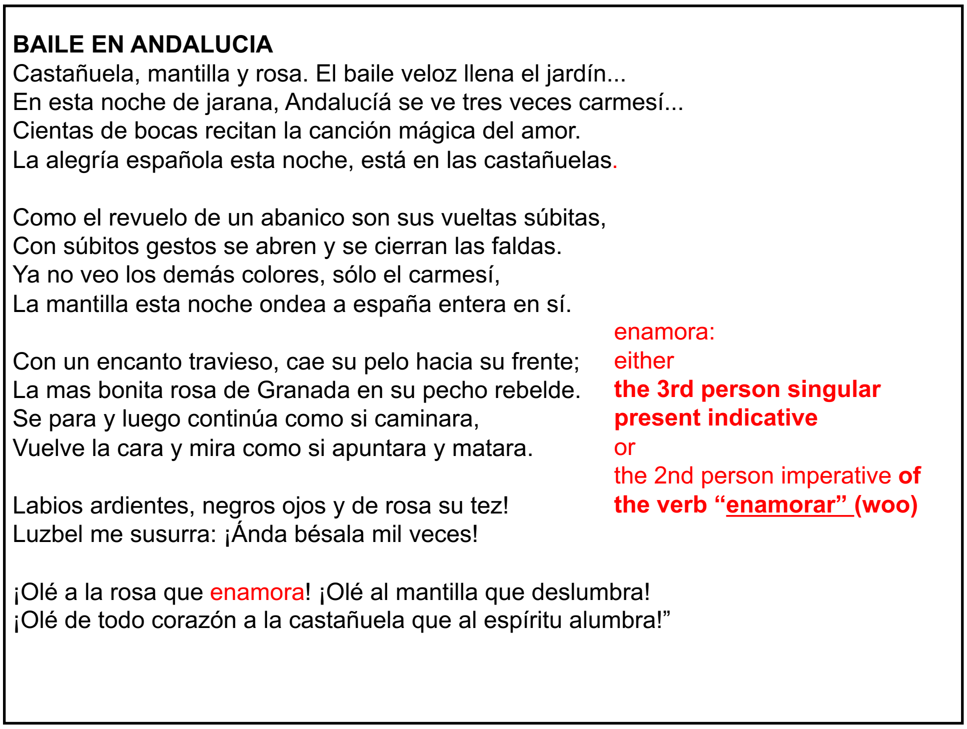
Como el revuelo de un abanico son sus vueltas súbitas,
Con súbitos gestos se abren y se cierran las faldas.
Ya no veo los demás colores, sólo el carmesí,
La mantilla esta noche ondea a españa entera en sí.

Con un encanto travieso, cae su pelo hacia su frente;
La mas bonita rosa de Granada en su pecho rebelde.
Se para y luego continúa como si caminara,
Vuelve la cara y mira como si apuntara y matara.

Labios ardientes, negros ojos y de rosa su tez!
Luzbel me susurra: ¡Ánda bésala mil veces!

¡Olé a la rosa que enamora! ¡Olé al mantilla que deslumbra!
¡Olé de todo corazón a la castañuela que al espíritu alumbra!"

veo:
First person present indicative of the verb "ver" (see?)



BAILE EN ANDALUCIA
Castañuela, mantilla y rosa. El baile veloz llena el jardín...
En esta noche de jarana, Andalucíá se ve tres veces carmesí...
Cientas de bocas recitan la canción mágica del amor.
La alegría española esta noche, está en las castañuelas.

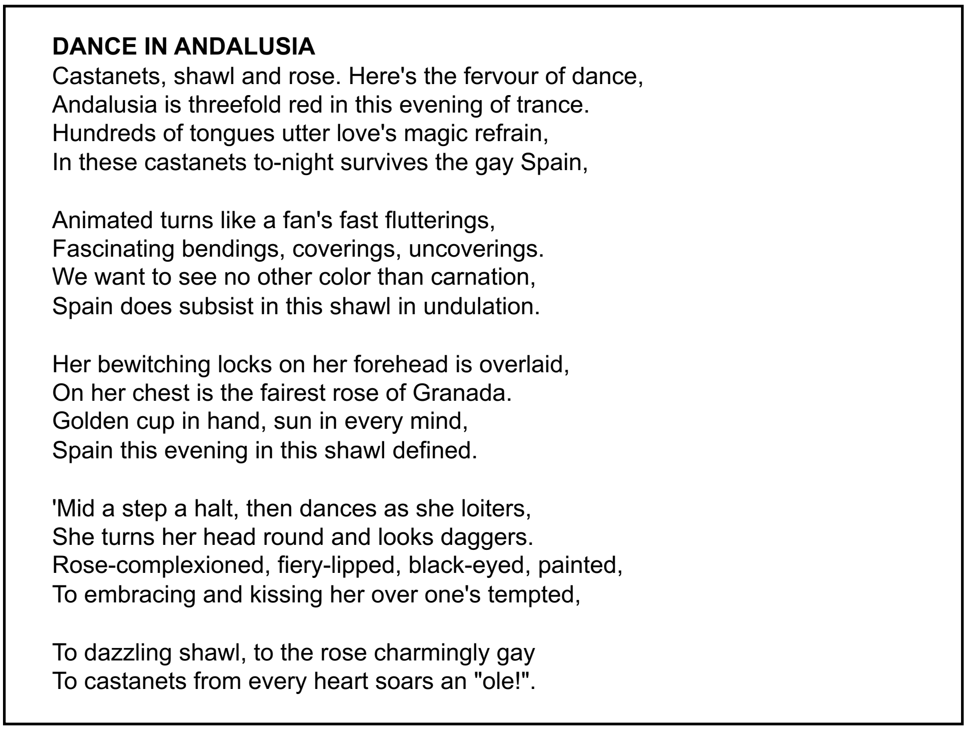
Como el revuelo de un abanico son sus vueltas súbitas,
Con súbitos gestos se abren y se cierran las faldas.
Ya no veo los demás colores, sólo el carmesí,
La mantilla esta noche ondea a españa entera en sí.

Con un encanto travieso, cae su pelo hacia su frente;
La mas bonita rosa de Granada en su pecho rebelde.
Se para y luego continúa como si caminara,
Vuelve la cara y mira como si apuntara y matara.

Labios ardientes, negros ojos y de rosa su tez!
Luzbel me susurra: ¡Ánda bésala mil veces!

¡Olé a la rosa que enamora! ¡Olé al mantilla que deslumbra!
¡Olé de todo corazón a la castañuela que al espíritu alumbra!"

enamora:
either
the 3rd person singular present indicative
or
the 2nd person imperative **of the verb "enamorar" (woo)**



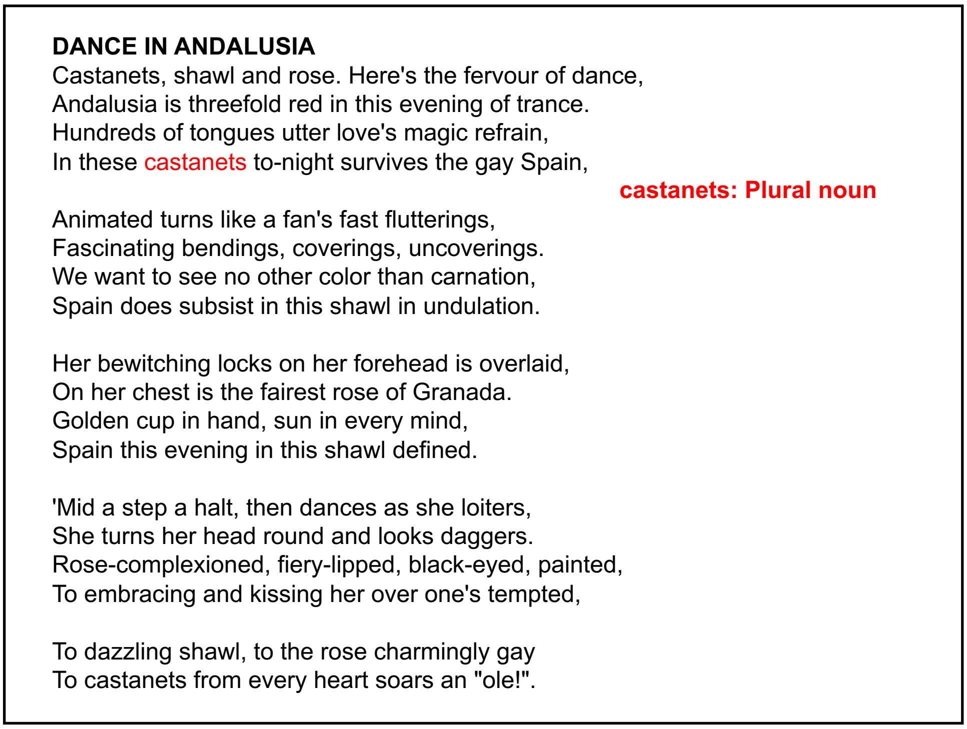
DANCE IN ANDALUSIA
Castanets, shawl and rose. Here's the fervour of dance,
Andalusia is threefold red in this evening of trance.
Hundreds of tongues utter love's magic refrain,
In these castanets to-night survives the gay Spain,

Animated turns like a fan's fast flutterings,
Fascinating bendings, coverings, uncoverings.
We want to see no other color than carnation,
Spain does subsist in this shawl in undulation.

Her bewitching locks on her forehead is overlaid,
On her chest is the fairest rose of Granada.
Golden cup in hand, sun in every mind,
Spain this evening in this shawl defined.

'Mid a step a halt, then dances as she loiters,
She turns her head round and looks daggers.
Rose-complexioned, fiery-lipped, black-eyed, painted,
To embracing and kissing her over one's tempted,

To dazzling shawl, to the rose charmingly gay
To castanets from every heart soars an "ole!".



DANCE IN ANDALUSIA
Castanets, shawl and rose. Here's the fervour of dance,
Andalusia is threefold red in this evening of trance.
Hundreds of tongues utter love's magic refrain,
In these castanets to-night survives the gay Spain,

castanets: Plural noun

Animated turns like a fan's fast flutterings,
Fascinating bendings, coverings, uncoverings.
We want to see no other color than carnation,
Spain does subsist in this shawl in undulation.

Her bewitching locks on her forehead is overlaid,
On her chest is the fairest rose of Granada.
Golden cup in hand, sun in every mind,
Spain this evening in this shawl defined.

'Mid a step a halt, then dances as she loiters,
She turns her head round and looks daggers.
Rose-complexioned, fiery-lipped, black-eyed, painted,
To embracing and kissing her over one's tempted,

To dazzling shawl, to the rose charmingly gay
To castanets from every heart soars an "ole!".

DANCE IN ANDALUSIA
Castanets, shawl and rose. Here's the fervour of dance,
Andalusia is threefold red in this evening of trance.
Hundreds of tongues utter love's magic refrain,
In these castanets to-night survives the gay Spain,

bewitching: gerund form of the verb bewitch

Animated turns like a fan's fast flutterings,
Fascinating bendings, coverings, uncoverings.
We want to see no other color than carnation,
Spain does subsist in this shawl in undulation.

Her bewitching locks on her forehead is overlaid,
On her chest is the fairest rose of Granada.
Golden cup in hand, sun in every mind,
Spain this evening in this shawl defined.

'Mid a step a halt, then dances as she loiters,
She turns her head round and looks daggers.
Rose-complexioned, fiery-lipped, black-eyed, painted,
To embracing and kissing her over one's tempted,

To dazzling shawl, to the rose charmingly gay
To castanets from every heart soars an "ole!".

DANCE IN ANDALUSIA
Castanets, shawl and rose. Here's the fervour of dance,
Andalusia is threefold red in this evening of trance.
Hundreds of tongues utter love's magic refrain,
In these castanets to-night survives the gay Spain,

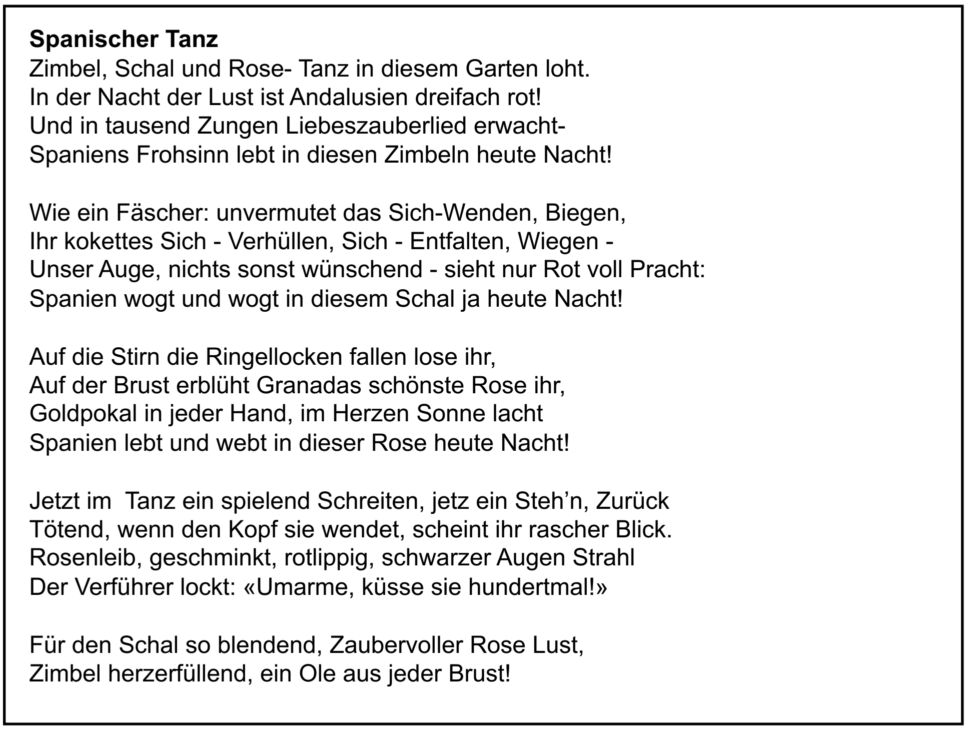
Animated turns like a fan's fast flutterings,
Fascinating bendings, coverings, uncoverings.
We want to see no other color than carnation,
Spain does subsist in this shawl in undulation.

Her bewitching locks on her forehead is overlaid,
On her chest is the fairest rose of Granada.
Golden cup in hand, sun in every mind,
Spain this evening in this shawl defined.

evening:
either
a noun
or
the present continuous
form of the verb “even”

'Mid a step a halt, then dances as she loiters,
She turns her head round and looks daggers.
Rose-complexioned, fiery-lipped, black-eyed, painted,
To embracing and kissing her over one's tempted,

To dazzling shawl, to the rose charmingly gay
To castanets from every heart soars an "ole!".



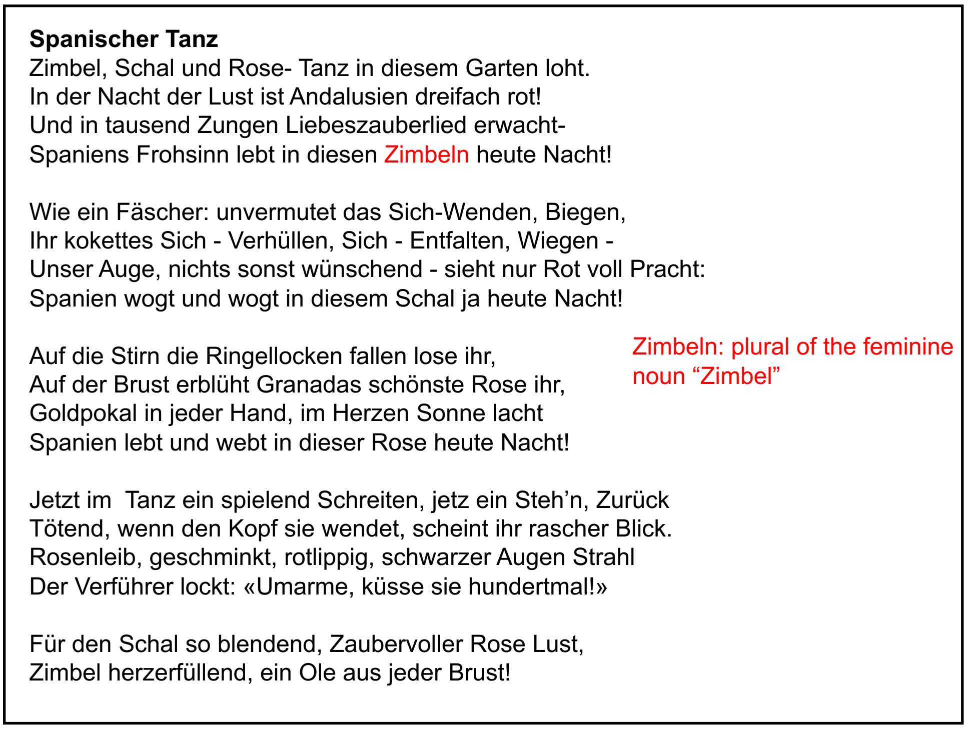
Spanischer Tanz
Zimbel, Schal und Rose- Tanz in diesem Garten loht.
In der Nacht der Lust ist Andalusien dreifach rot!
Und in tausend Zungen Liebeszauberlied erwacht-
Spaniens Frohsinn lebt in diesen Zimbeln heute Nacht!

Wie ein Fäscher: unvermutet das Sich-Wenden, Biegen,
Ihr kokettes Sich - Verhüllen, Sich - Entfalten, Wiegen -
Unser Auge, nichts sonst wünschend - sieht nur Rot voll Pracht:
Spanien wogt und wogt in diesem Schal ja heute Nacht!

Auf die Stirn die Ringellocken fallen lose ihr,
Auf der Brust erblüht Granadas schönste Rose ihr,
Goldpokal in jeder Hand, im Herzen Sonne lacht
Spanien lebt und webt in dieser Rose heute Nacht!

Jetzt im Tanz ein spielend Schreiten, jetz ein Steh'n, Zurück
Tötend, wenn den Kopf sie wendet, scheint ihr rascher Blick.
Rosenleib, geschminkt, rotlippig, schwarzer Augen Strahl
Der Verführer lockt: «Umarme, küsse sie hundertmal!»

Für den Schal so blendend, Zaubervoller Rose Lust,
Zimbel herzerfüllend, ein Ole aus jeder Brust!



Spanischer Tanz
Zimbel, Schal und Rose- Tanz in diesem Garten loht.
In der Nacht der Lust ist Andalusien dreifach rot!
Und in tausend Zungen Liebeszauberlied erwacht-
Spaniens Frohsinn lebt in diesen Zimbeln heute Nacht!

Wie ein Fäscher: unvermutet das Sich-Wenden, Biegen,
Ihr kokettes Sich - Verhüllen, Sich - Entfalten, Wiegen -
Unser Auge, nichts sonst wünschend - sieht nur Rot voll Pracht:
Spanien wogt und wogt in diesem Schal ja heute Nacht!

Auf die Stirn die Ringellocken fallen lose ihr,
Auf der Brust erblüht Granadas schönste Rose ihr,
Goldpokal in jeder Hand, im Herzen Sonne lacht
Spanien lebt und webt in dieser Rose heute Nacht!

Zimbeln: plural of the feminine noun “Zimbel”

Jetzt im Tanz ein spielend Schreiten, jetz ein Steh'n, Zurück
Tötend, wenn den Kopf sie wendet, scheint ihr rascher Blick.
Rosenleib, geschminkt, rotlippig, schwarzer Augen Strahl
Der Verführer lockt: «Umarme, küsse sie hundertmal!»

Für den Schal so blendend, Zaubervoller Rose Lust,
Zimbel herzerfüllend, ein Ole aus jeder Brust!

Spanischer Tanz
Zimbel, Schal und Rose- Tanz in diesem Garten loht.
In der Nacht der Lust ist Andalusien dreifach rot!
Und in tausend Zungen Liebeszauberlied erwacht-
Spaniens Frohsinn lebt in diesen Zimbeln heute Nacht!

Wie ein Fäscher: unvermutet das Sich-Wenden, Biegen,
Ihr kokettes Sich - Verhüllen, Sich - Entfalten, Wiegen -
Unser Auge, nichts sonst wünschend - sieht nur Rot voll Pracht:
Spanien wogt und wogt in diesem Schal ja heute Nacht!

Auf die Stirn die Ringellocken fallen lose ihr,
Auf der Brust erblüht Granadas schönste Rose ihr,
Goldpokal in jeder Hand, im Herzen Sonne lacht
Spanien lebt und webt in dieser Rose heute Nacht!

Liebeszauberlied: compound noun
Magic love song (?)

Jetzt im Tanz ein spielend Schreiten, jetz ein Steh'n, Zurück
Tötend, wenn den Kopf sie wendet, scheint ihr rascher Blick.
Rosenleib, geschminkt, rotlippig, schwarzer Augen Strahl
Der Verführer lockt: «Umarme, küsse sie hundertmal!»

Für den Schal so blendend, Zaubervoller Rose Lust,
Zimbel herzerfüllend, ein Ole aus jeder Brust!

Spanischer Tanz
Zimbel, Schal und Rose- Tanz in diesem Garten loht.
In der Nacht der Lust ist Andalusien dreifach rot!
Und in tausend Zungen Liebeszauberlied erwacht-
Spaniens Frohsinn lebt in diesen Zimbeln heute Nacht!

Wie ein Fäscher: unvermutet das Sich-Wenden, Biegen,
Ihr kokettes Sich - Verhüllen, Sich - Entfalten, Wiegen -
Unser Auge, nichts sonst wünschend - sieht nur Rot voll Pracht:
Spanien wogt und wogt in diesem Schal ja heute Nacht!

Auf die Stirn die Ringellocken fallen lose ihr,
Auf der Brust erblüht Granadas schönste Rose ihr,
Goldpokal in jeder Hand, im Herzen Sonne lacht
Spanien lebt und webt in dieser Rose heute Nacht!

Ringellocken: compound noun
Convoluted curls (?)

Jetzt im Tanz ein spielend Schreiten, jetz ein Steh'n, Zurück
Tötend, wenn den Kopf sie wendet, scheint ihr rascher Blick.
Rosenleib, geschminkt, rotlippig, schwarzer Augen Strahl
Der Verführer lockt: «Umarme, küsse sie hundertmal!»

Für den Schal so blendend, Zaubervoller Rose Lust,
Zimbel herzerfüllend, ein Ole aus jeder Brust!

Spanischer Tanz

Zimbel, Schal und Rose- Tanz in diesem Garten loht.
In der Nacht der Lust ist Andalusien dreifach rot!
Und in tausend Zungen Liebeszauberlied erwacht-
Spaniens Frohsinn lebt in diesen Zimbeln heute Nacht!

Wie ein Fäscher: unvermutet das Sich-Wenden, Biegen,
Ihr kokettes Sich - Verhüllen, Sich - Entfalten, Wiegen -
Unser Auge, nichts sonst wünschend - sieht nur Rot voll Pracht:
Spanien wogt und wogt in diesem Schal ja heute Nacht!

Auf die Stirn die Ringellocken fallen lose ihr,
Auf der Brust erblüht Granadas schönste Rose ihr,
Goldpokal in jeder Hand, im Herzen Sonne lacht
Spanien lebt und webt in dieser Rose heute Nacht!

herzerfüllend: noun-verb/participle compound
“that which fulfills the heart”(?)

Jetzt im Tanz ein spielend Schreiten, jetz ein Steh’n, Zurück
Tötend, wenn den Kopf sie wendet, scheint ihr rascher Blick.
Rosenleib, geschminkt, rotlippig, schwarzer Augen Strahl
Der Verführer lockt: «Umarme, küsse sie hundertmal!»

Für den Schal so blendend, Zaubervoller Rose Lust,
Zimbel herzerfüllend, ein Ole aus jeder Brust!

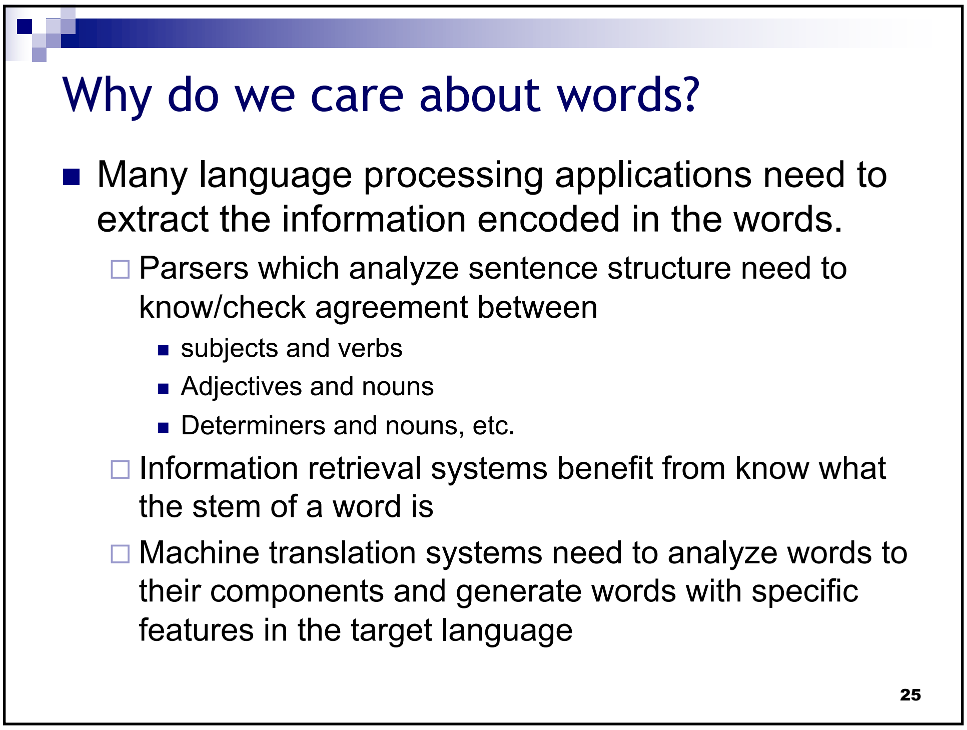
Aligned Verses

Zil, şal ve gül, bu bahçede raksın bütün hızı
Şevk akşamında Endülüs, üç defa kırmızı
Aşkın sihirli şarkısı, yüzlerce dildedir
İspanya neş'esiyle bu akşam bu zildedir

Castañuela, mantilla y rosa. El baile veloz llena el jardín...
En esta noche de jarana, Andalucíá se ve tres veces carmesí...
Cientas de bocas recitan la canción mágica del amor.
La alegría española esta noche, está en las castañuelas.

Castanets, shawl and rose. Here's the fervour of dance,
Andalusia is threefold red in this evening of trance.
Hundreds of tongues utter love's magic refrain,
In these castanets to-night survives the gay Spain,

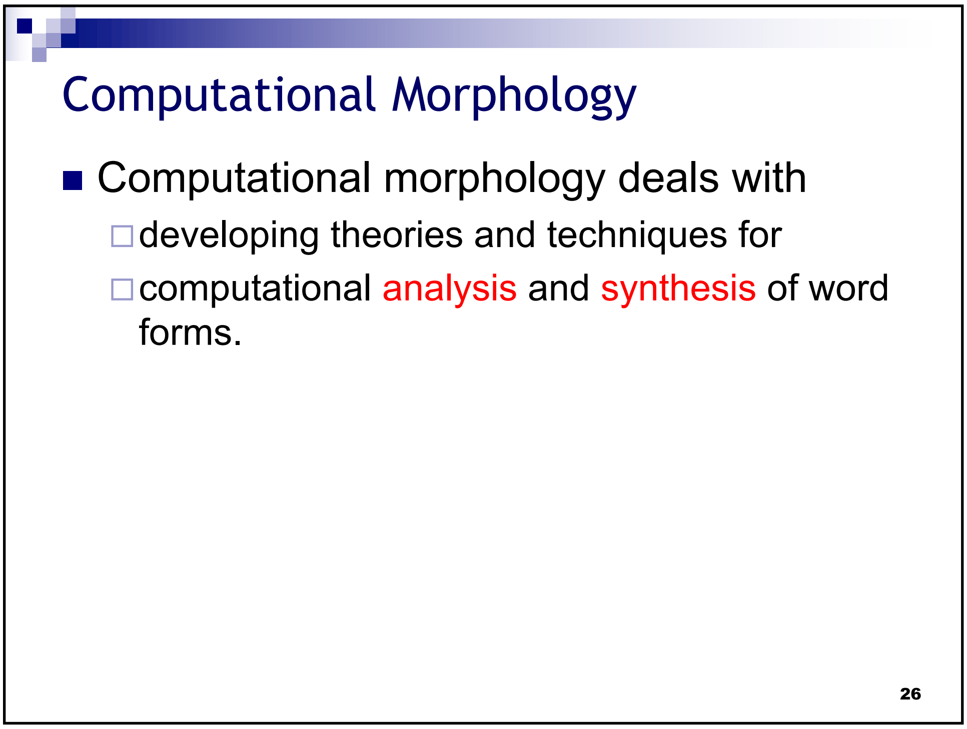
Zimbel, Schal und Rose- Tanz in diesem Garten loht.
In der Nacht der Lust ist Andalusien dreifach rot!
Und in tausend Zungen Liebeszauberlied erwacht-
Spaniens Frohsinn lebt in diesen Zimbeln heute Nacht!



Why do we care about words?

- Many language processing applications need to extract the information encoded in the words.
  - □ Parsers which analyze sentence structure need to know/check agreement between
    - subjects and verbs
    - Adjectives and nouns
    - Determiners and nouns, etc.
  - □ Information retrieval systems benefit from know what the stem of a word is
  - □ Machine translation systems need to analyze words to their components and generate words with specific features in the target language

25



Computational Morphology

- Computational morphology deals with
  - □ developing theories and techniques for
  - □ computational analysis and synthesis of word forms.

26

Computational Morphology -Analysis

- Extract any information encoded in a word and bring it out so that later layers of processing can make use of it.

- books ⇒ book+Noun+Plural
  ⇒ book+Verb+Pres+3SG.
- stopping ⇒ stop+Verb+Cont
- happiest ⇒ happy+Adj+Superlative
- went ⇒ go+Verb+Past

27

Computational Morphology -Generation

- In a machine translation application, one may have to generate the word corresponding to a set of features

- stop+Past ⇒ stopped
- (T) dur+Past+1Pl ⇒ durduk
  +2Pl ⇒ durdunuz

28

Computational Morphology-Analysis

- Input raw text
- Segment / Tokenize

} Pre-processing

- Analyze individual words
- Analyze multi-word constructs
- Disambiguate Morphology

} Morphological Processing

- Syntactically analyze sentences

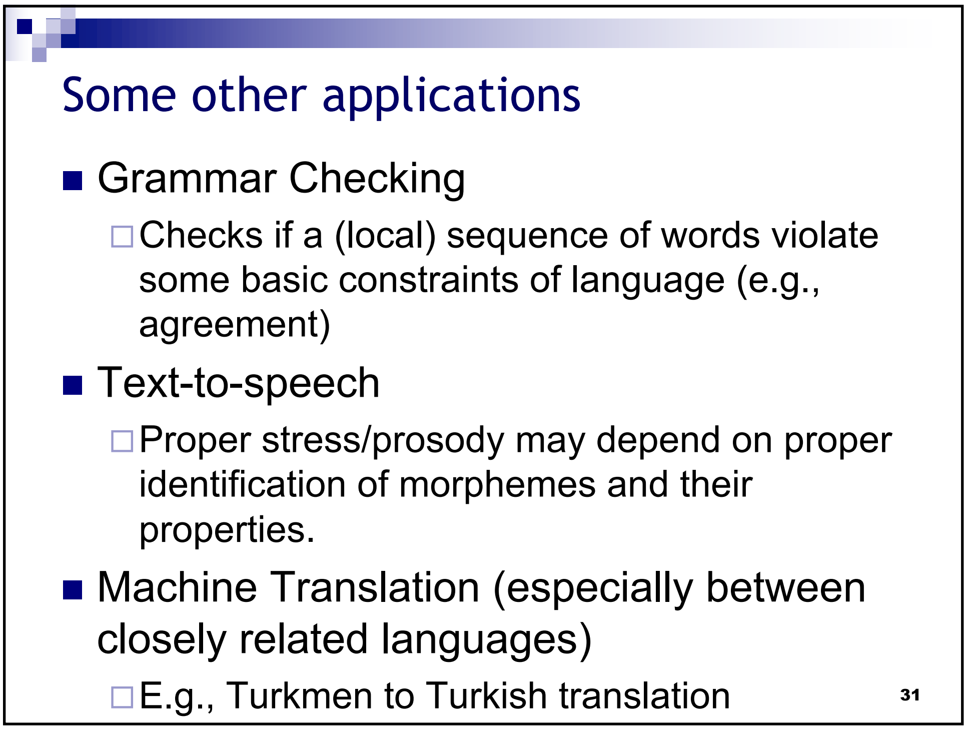
} Syntactic Processing

- ….

Some other applications

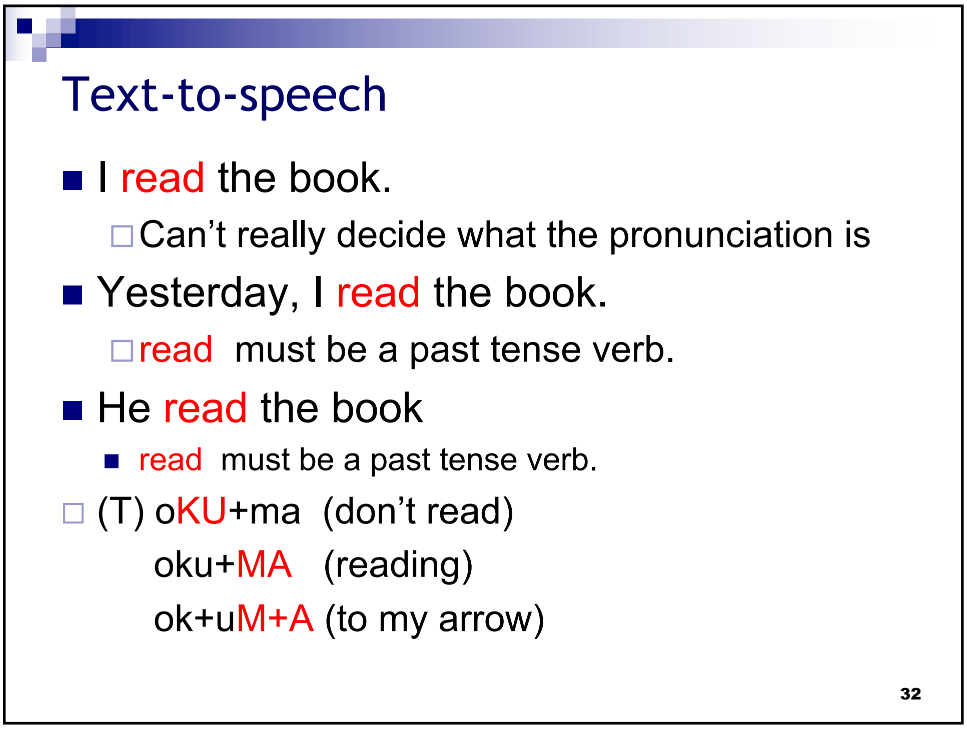
- Spelling Checking
  - □ Check if words in a text are all valid words
- Spelling Correction
  - □ Find correct words “close” to a misspelled word.

- For both these applications, one needs to know what constitutes a valid word in a language.
  - □ Rather straightforward for English
  - □ No so for Turkish –
    - cezalandırılacaklardan (ceza+lan+dır+ıl+acak+lar+dan)



Some other applications

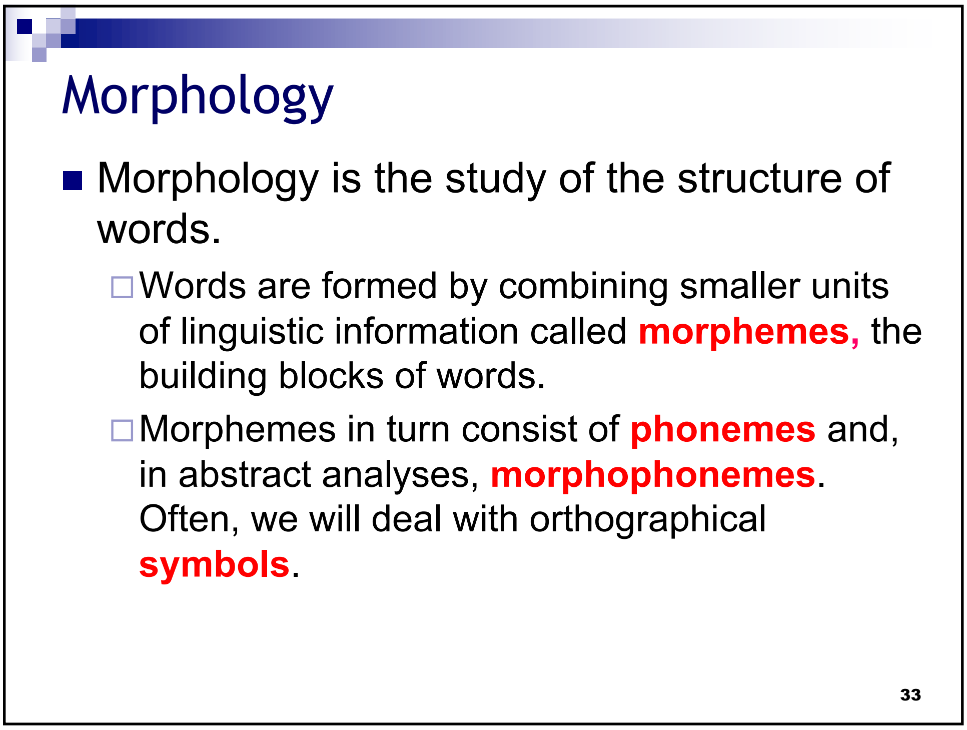
- Grammar Checking
  - Checks if a (local) sequence of words violate some basic constraints of language (e.g., agreement)
- Text-to-speech
  - Proper stress/prosody may depend on proper identification of morphemes and their properties.
- Machine Translation (especially between closely related languages)
  - E.g., Turkmen to Turkish translation



Text-to-speech

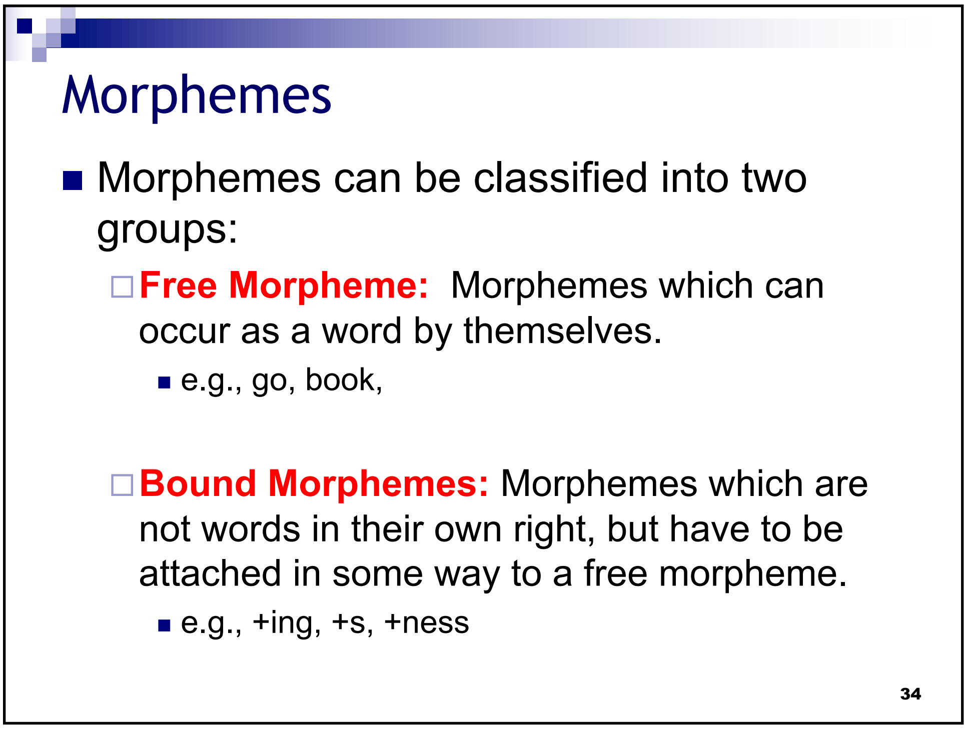
- I read the book.
  - Can't really decide what the pronunciation is
- Yesterday, I read the book.
  - read must be a past tense verb.
- He read the book
  - read must be a past tense verb.

□ (T) oKU+ma (don't read)
oku+MA   (reading)
ok+uM+A (to my arrow)



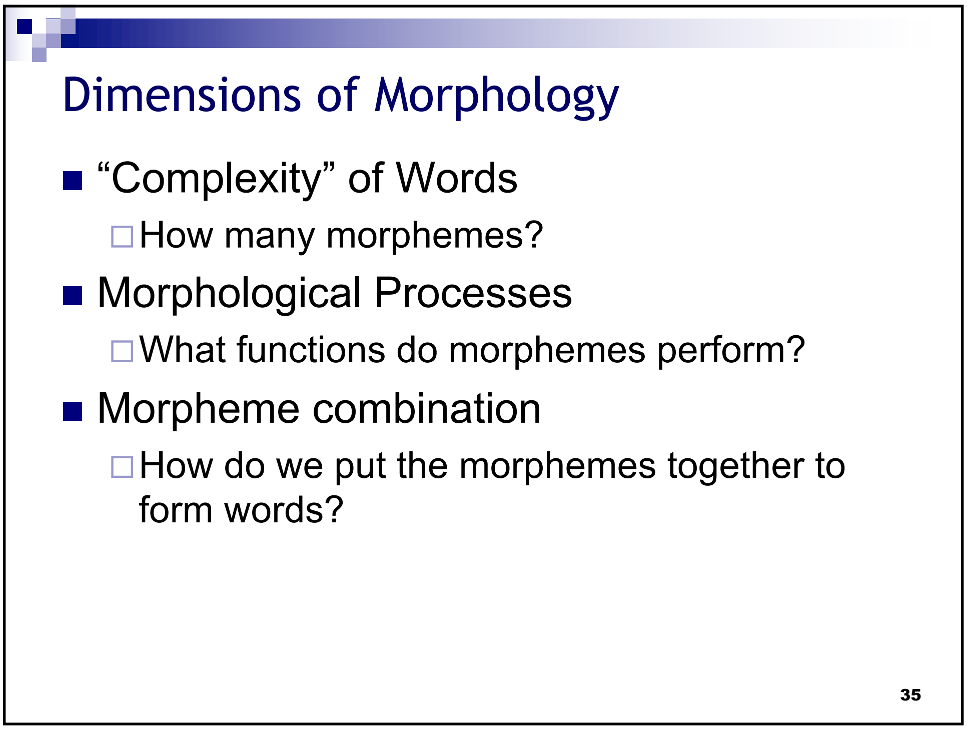
Morphology

- Morphology is the study of the structure of words.
  - □ Words are formed by combining smaller units of linguistic information called **morphemes,** the building blocks of words.
  - □ Morphemes in turn consist of **phonemes** and, in abstract analyses, **morphophonemes**. Often, we will deal with orthographical **symbols**.



Morphemes

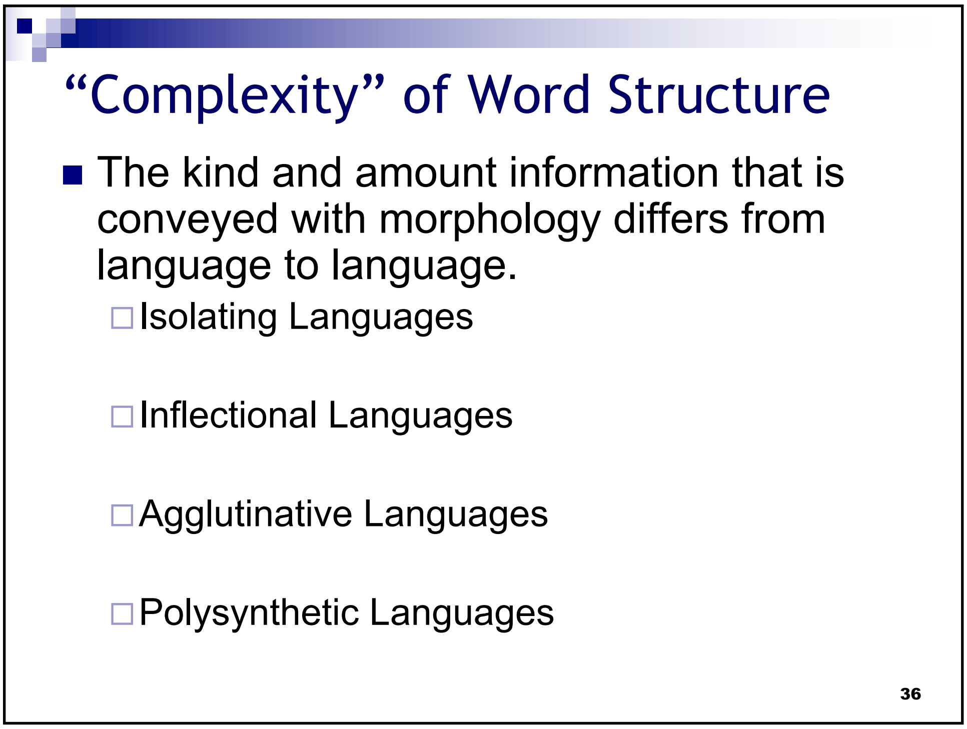
- Morphemes can be classified into two groups:
  - □ **Free Morpheme:** Morphemes which can occur as a word by themselves.
    - e.g., go, book,
  - □ **Bound Morphemes:** Morphemes which are not words in their own right, but have to be attached in some way to a free morpheme.
    - e.g., +ing, +s, +ness



Dimensions of Morphology

- “Complexity” of Words
  - □ How many morphemes?
- Morphological Processes
  - □ What functions do morphemes perform?
- Morpheme combination
  - □ How do we put the morphemes together to form words?

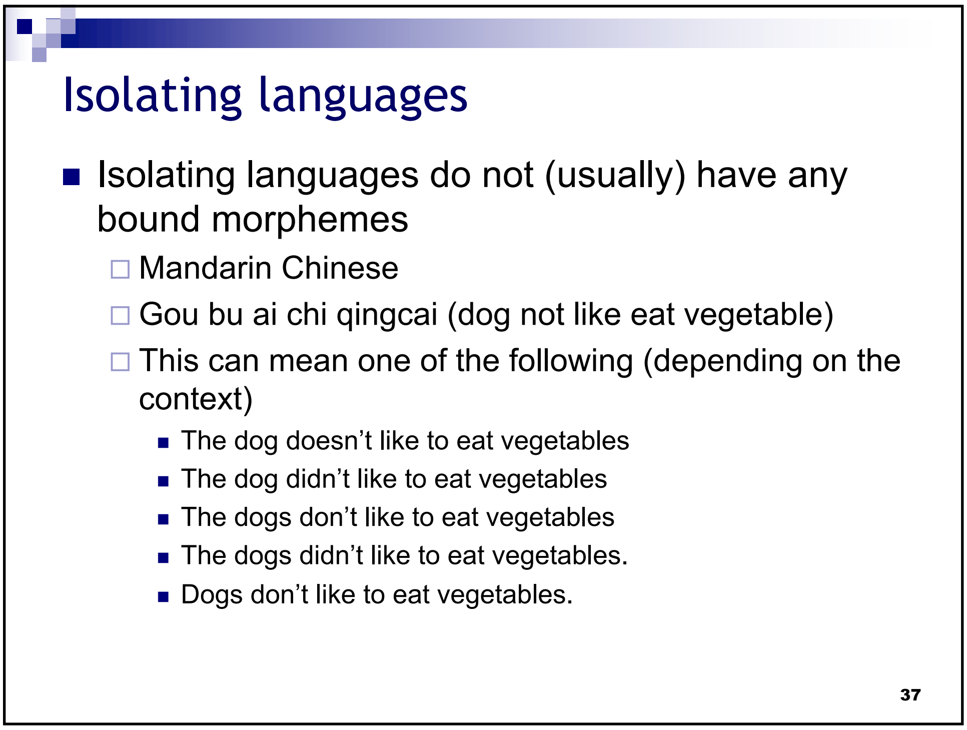
35



“Complexity” of Word Structure

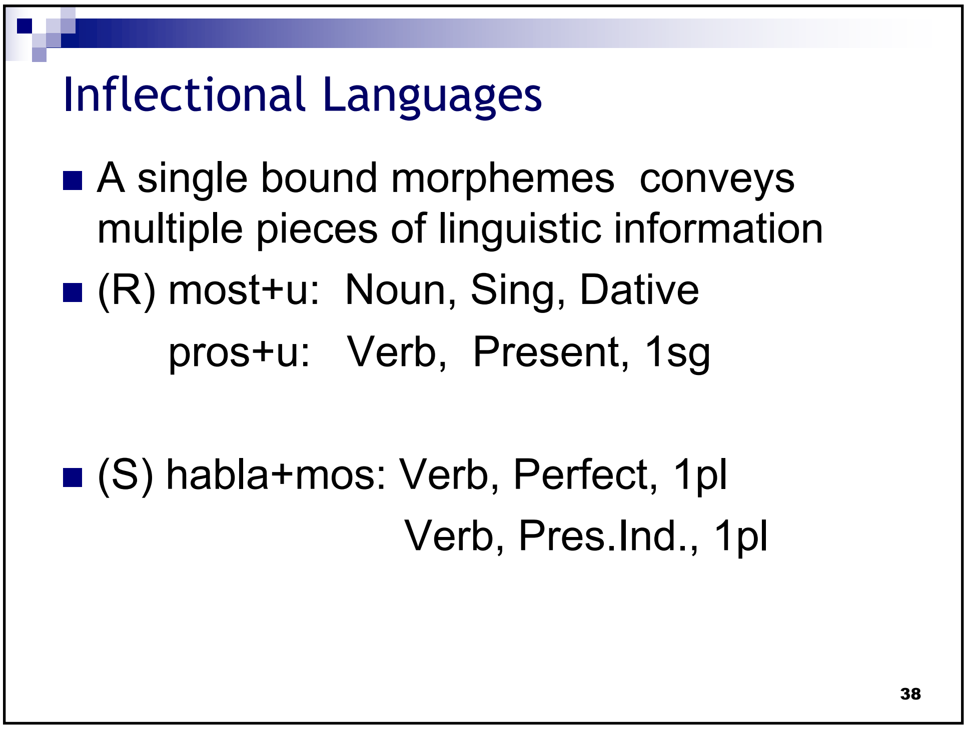
- The kind and amount information that is conveyed with morphology differs from language to language.
  - □ Isolating Languages
  - □ Inflectional Languages
  - □ Agglutinative Languages
  - □ Polysynthetic Languages

36



Isolating languages

- Isolating languages do not (usually) have any bound morphemes
  - □ Mandarin Chinese
  - □ Gou bu ai chi qingcai (dog not like eat vegetable)
  - □ This can mean one of the following (depending on the context)
    - The dog doesn’t like to eat vegetables
    - The dog didn’t like to eat vegetables
    - The dogs don’t like to eat vegetables
    - The dogs didn’t like to eat vegetables.
    - Dogs don’t like to eat vegetables.



Inflectional Languages

- A single bound morphemes conveys multiple pieces of linguistic information
- (R) most+u: Noun, Sing, Dative

  pros+u: Verb, Present, 1sg

- (S) habla+mos: Verb, Perfect, 1pl

  Verb, Pres.Ind., 1pl

Agglutinative Languages

- (Usually multiple) Bound morphemes are attached to one (or more) free morphemes, like beads on a string.
  - □ Turkish/Turkic, Finnish, Hungarian
  - □ Swahili, Aymara
- Each morpheme encodes one "piece" of linguistic information.
  - □ (T) gid+iyor+du+m: continuous, Past, 1sg (I was going)

Agglutinative Languages

- Turkish
- Finlandiyalılaştıramadıklarımızdanmışsınızcasına
  - (behaving) as if you have been one of those whom we could not convert into a Finn(ish citizen)/someone from Finland
  - Finlandiya+lı+laş+tır+ama+dık+lar+ımız+dan+mış+sınız+casına
- □ Finlandiya+Noun+Prop+A3sg+Pnon+Nom
  - ^DB+Adj+With/From
  - ^DB+Verb+Become
  - ^DB+Verb+Caus
  - ^DB+Verb+Able+Neg
  - ^DB+Noun+PastPart+A3pl+P1pl+Abl
  - ^DB+Verb+Zero+Narr+A2pl
  - ^DB+Adverb+AsIf

Agglutinative Languages

- Aymara
  - □ ch'uñüwinkaskirïyätwa
  - □ ch'uñu +: +wi +na -ka +si -ka -iri +: +ya:t(a) +wa
- I was (one who was) always at the place for making ch'uñu'

ch'uñu	N	'freeze-dried potatoes'	
	+:	N>V	be/make …
	+wi	V>N	place-of
	+na		in (location)
	-ka	N>V	be-in (location)
	+si		continuative
	-ka		imperfect
	-iri	V>N	one who
	+:	N>V	be
	+ya:ta	1P	recent past
	+wa		affirmative sentencial

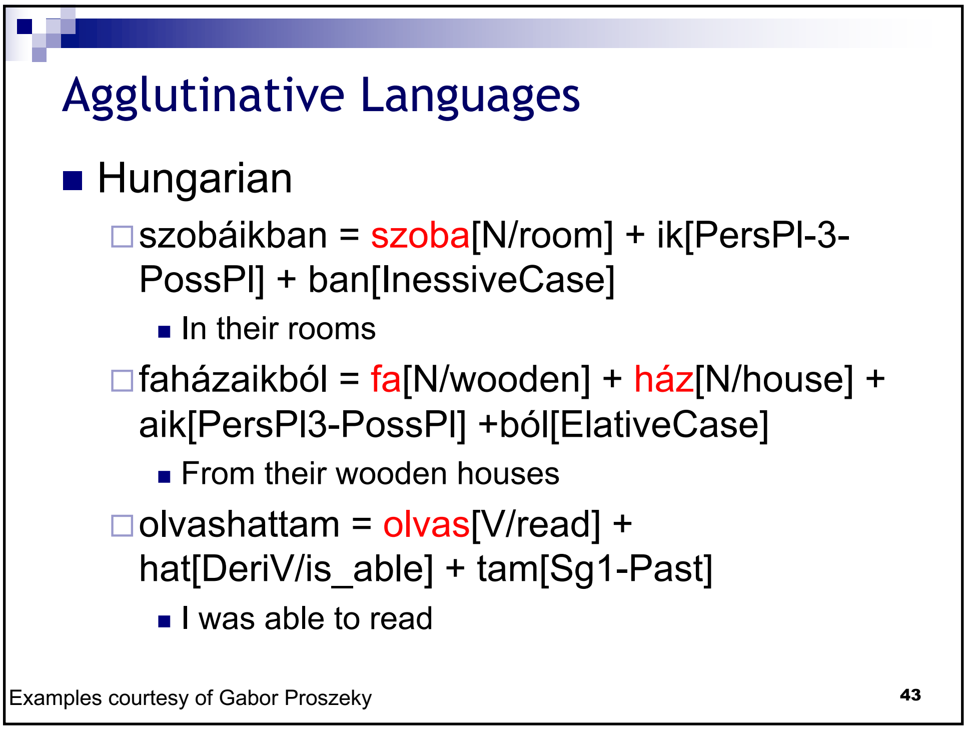
Example Courtesy of Ken Beesley

Agglutinative Languages

- Finnish Numerals
  - □ Finnish numerals are written as one word and all components inflect and agree in all aspects
  - □ **Kahdensienkymmenensienkahdeksansien**

two	ten	eighth (28th)
kaksi+Ord+Pl+Gen	kymmenen+Ord+Pl+Gen	kahdeksan+Ord+Pl+Gen
kahde ns i en	**kymmene ns i en**	**kahdeksa ns i en**

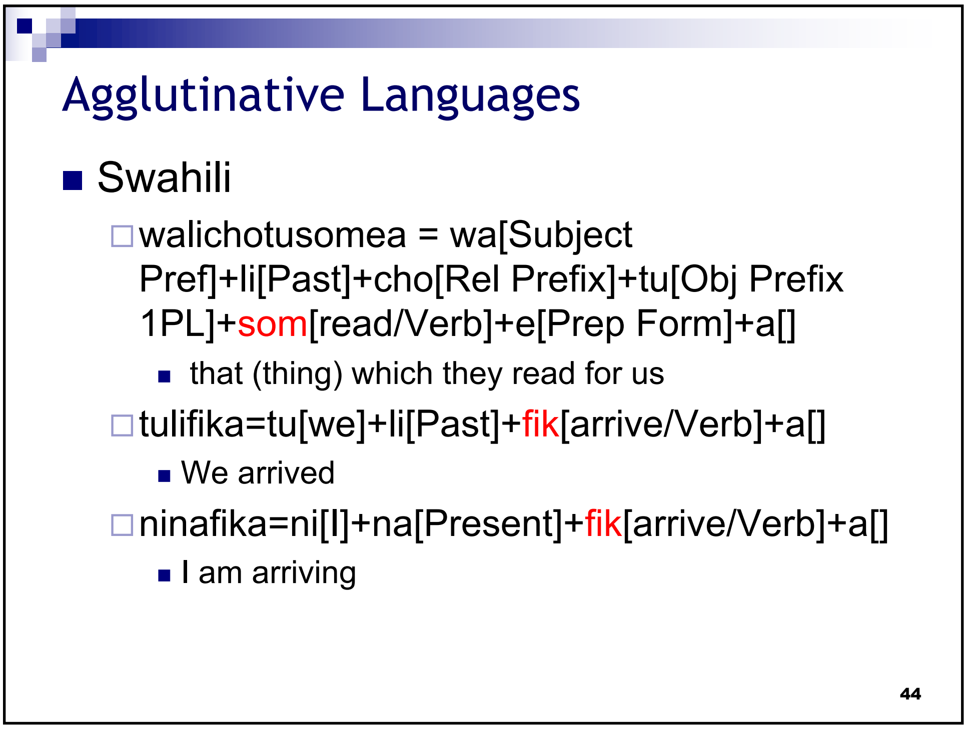
Example Courtesy of Lauri Karttunen



Agglutinative Languages

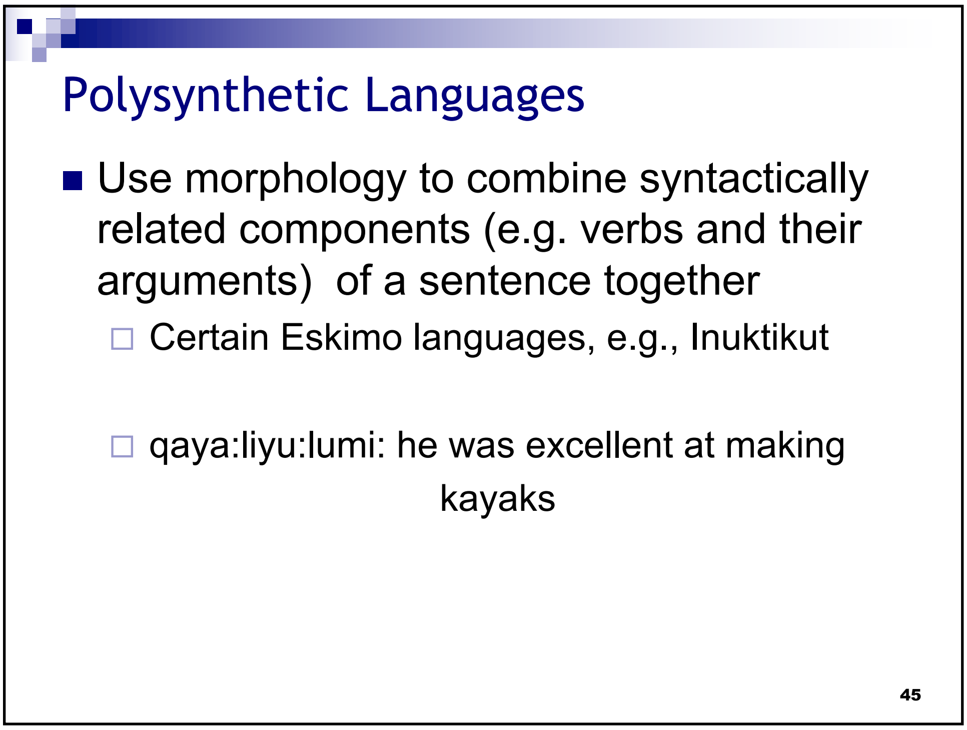
- Hungarian
  - □ szobáikban = szoba[N/room] + ik[PersPl-3-PossPl] + ban[InessiveCase]
    - In their rooms
  - □ faházaikból = fa[N/wooden] + ház[N/house] + aik[PersPl3-PossPl] +ból[ElativeCase]
    - From their wooden houses
  - □ olvashattam = olvas[V/read] + hat[DeriV/is_able] + tam[Sg1-Past]
    - I was able to read

Examples courtesy of Gabor Proszeky



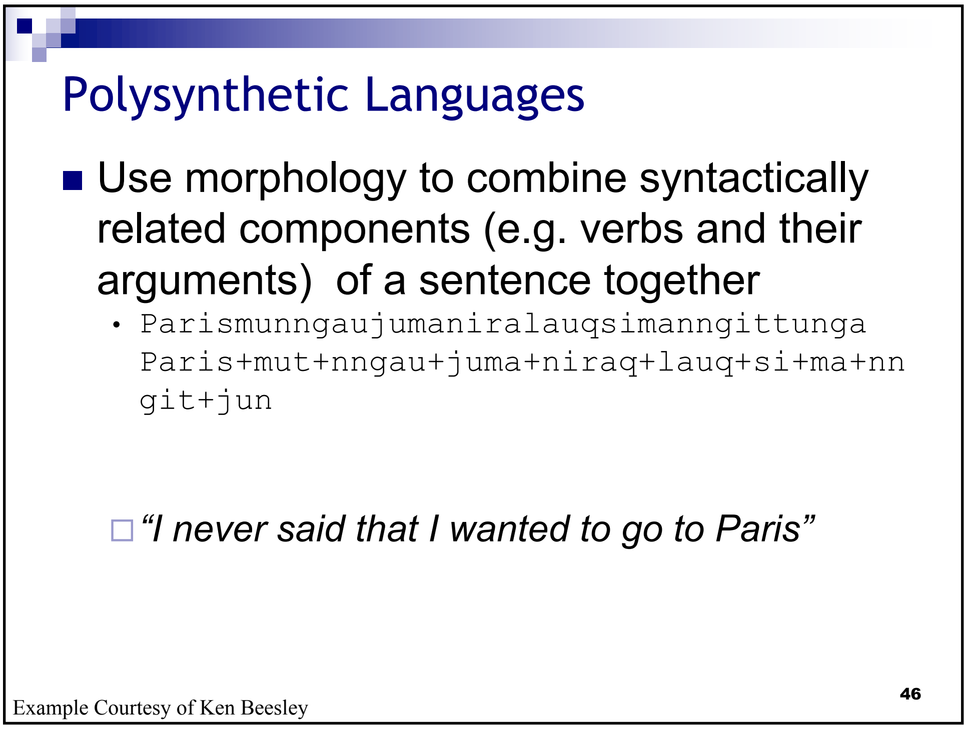
Agglutinative Languages

- Swahili
  - □ walichotusomea = wa[Subject Pref]+li[Past]+cho[Rel Prefix]+tu[Obj Prefix 1PL]+som[read/Verb]+e[Prep Form]+a[]
    - that (thing) which they read for us
  - □ tulifika=tu[we]+li[Past]+fik[arrive/Verb]+a[]
    - We arrived
  - □ ninafika=ni[I]+na[Present]+fik[arrive/Verb]+a[]
    - I am arriving



Polysynthetic Languages

- Use morphology to combine syntactically related components (e.g. verbs and their arguments) of a sentence together
  - □ Certain Eskimo languages, e.g., Inuktikut
  - □ qaya:liyu:lumi: he was excellent at making kayaks



Polysynthetic Languages

- Use morphology to combine syntactically related components (e.g. verbs and their arguments) of a sentence together
  - `Parismunngaujumaniralauqsimanngittunga`
    `Paris+mut+nngau+juma+niraq+lauq+si+ma+nngit+jun`
  - □ *"I never said that I wanted to go to Paris"*

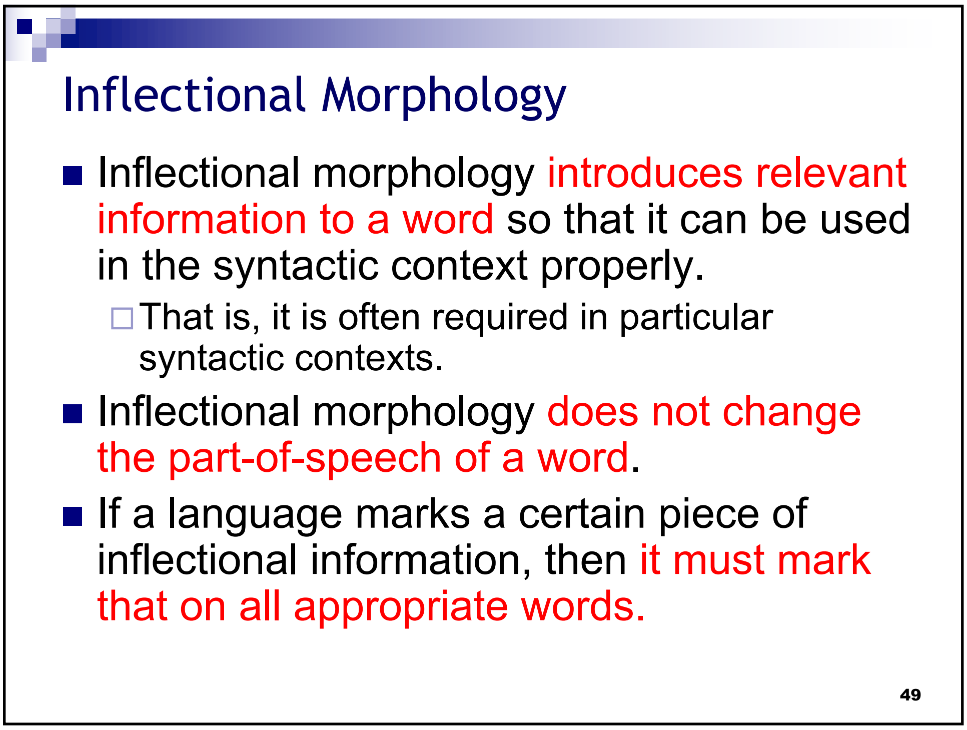
Example Courtesy of Ken Beesley

Arabic

- Arabic seems to have aspects of
  - □ Inflecting languages
    - wktbt (wa+katab+at "and she wrote …")
  - □ Agglutinative languages
    - wsyktbunha (wa+sa+ya+ktub+ūn+ha "and will (masc) they write her)
  - □ Polysynthetic languages

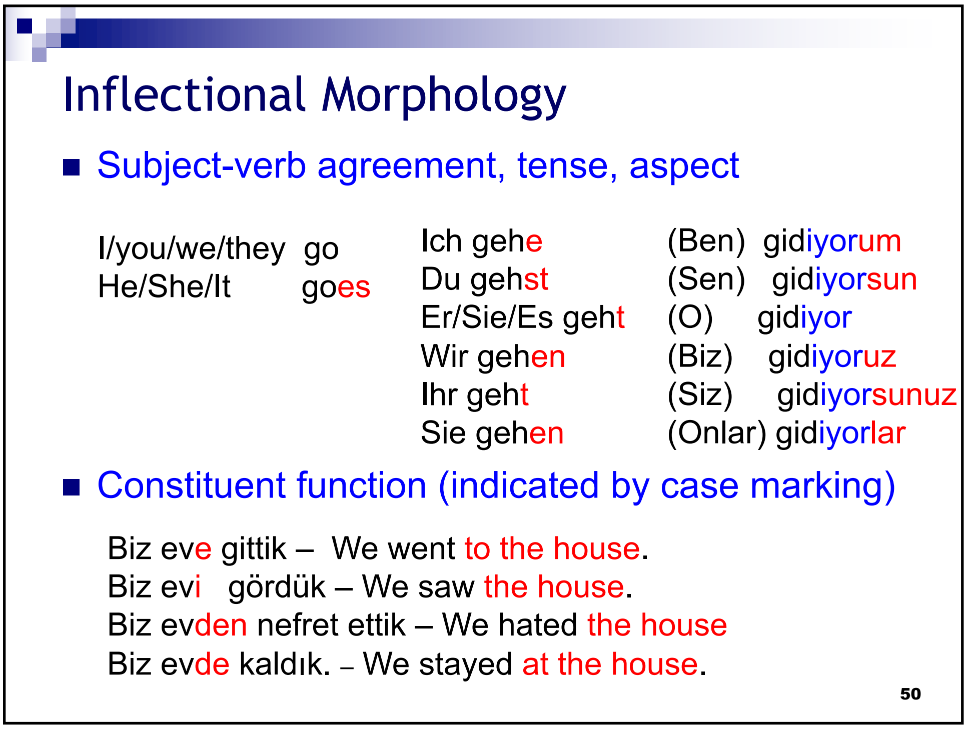
Morphological Processes

- There are essentially 3 types of morphological processes which determine the functions of morphemes:
  - □ Inflectional Morphology
  - □ Derivational Morphology
  - □ Compounding



Inflectional Morphology

- Inflectional morphology introduces relevant information to a word so that it can be used in the syntactic context properly.
  - □ That is, it is often required in particular syntactic contexts.
- Inflectional morphology does not change the part-of-speech of a word.
- If a language marks a certain piece of inflectional information, then it must mark that on all appropriate words.



Inflectional Morphology

- Subject-verb agreement, tense, aspect

I/you/we/they	go	Ich	gehe	(Ben)	gidiyorum
He/She/It	goes	Du	gehst	(Sen)	gidiyorsun
		Er/Sie/Es	geht	(O)	gidiyor
		Wir	gehen	(Biz)	gidiyoruz
		Ihr	geht	(Siz)	gidiyorsunuz
		Sie	gehen	(Onlar)	gidiyorlar

- Constituent function (indicated by case marking)

Biz eve gittik – We went to the house.
Biz evi gördük – We saw the house.
Biz evden nefret ettik – We hated the house
Biz evde kaldık. – We stayed at the house.

Inflectional Morphology

- Number, case, possession, gender, noun-class for nouns
  - □ (T) ev+ler+in+den (from your houses)
  - □ Bantu marks noun class by a prefix.
    - Humans: m+tu (person) wa+tu (persons)
    - Thin-objects: m+ti (tree) mi+ti (trees)
    - Paired things: ji-cho (eye) ma+cho (eyes)
    - Instrument: ki+tu (thing) vi+tu (things)
    - Extended body parts: u+limi (tongue) n+dimi (tongues)

Inflectional Morphology

- Gender and/or case marking may also appear on adjectives in agreement with the nouns they modify

(G) ein neu**er** Wagen
eine schön**e** Stadt
ein alt**es** Auto

Inflectional Morphology

- Case/Gender agreement for determiners

- (G) Der Bleistift (the pencil)
  Den Blestift (the pencil (object/Acc))
  Dem Bleistift (the pencil (Dative))
  Des Bleistifts (of the pencil)

  Die Frau (the woman)
  Die Frau (the woman (object(Acc))
  Der Frau (the woman (Dative)
  Der Frau (of the woman)

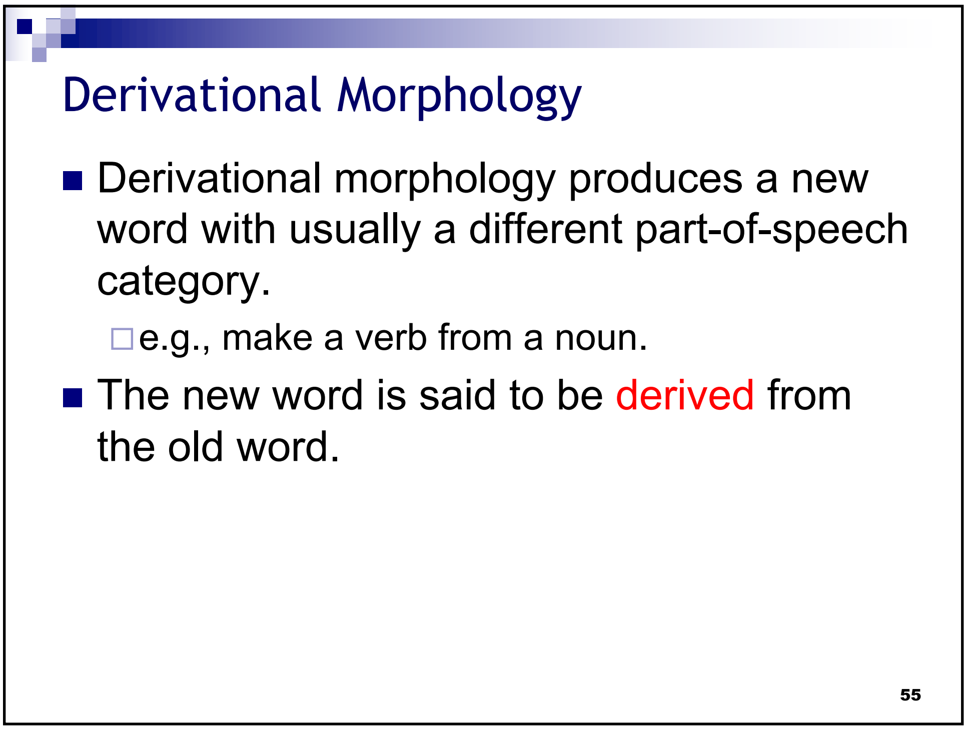
Inflectional Morphology

- (A) Perfect verb subject conjugation (masc form only)

Singular	Dual	Plural
katabtu		katabnā
katabta	katabtumā	katabtum
kataba	katabā	katabtū

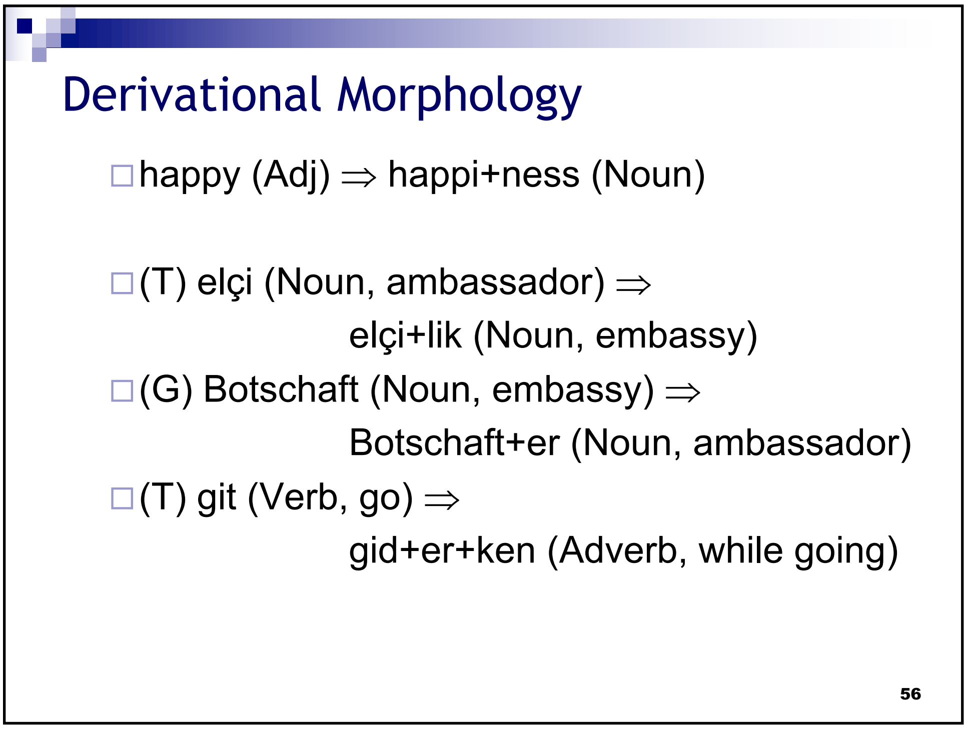
- (A) Imperfect verb subject conjugation

Singular	Dual	Plural
aktubu		naktubu
taktubu	taktubān	taktubūn
yaktubu	yaktubān	yaktubūn



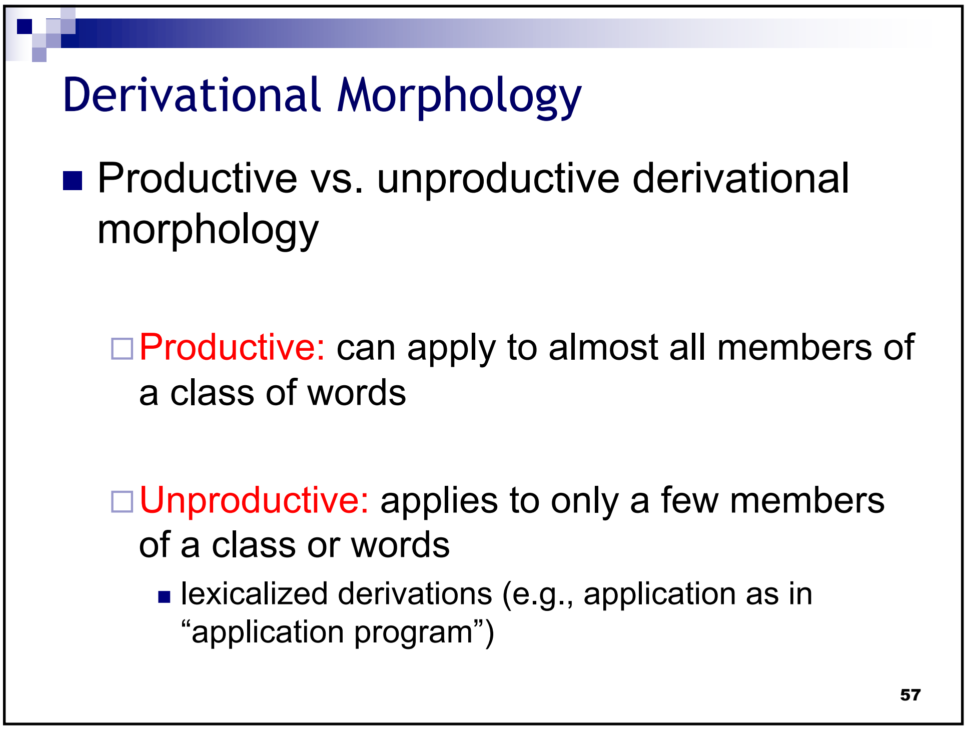
Derivational Morphology

- Derivational morphology produces a new word with usually a different part-of-speech category.
  - □e.g., make a verb from a noun.
- The new word is said to be derived from the old word.



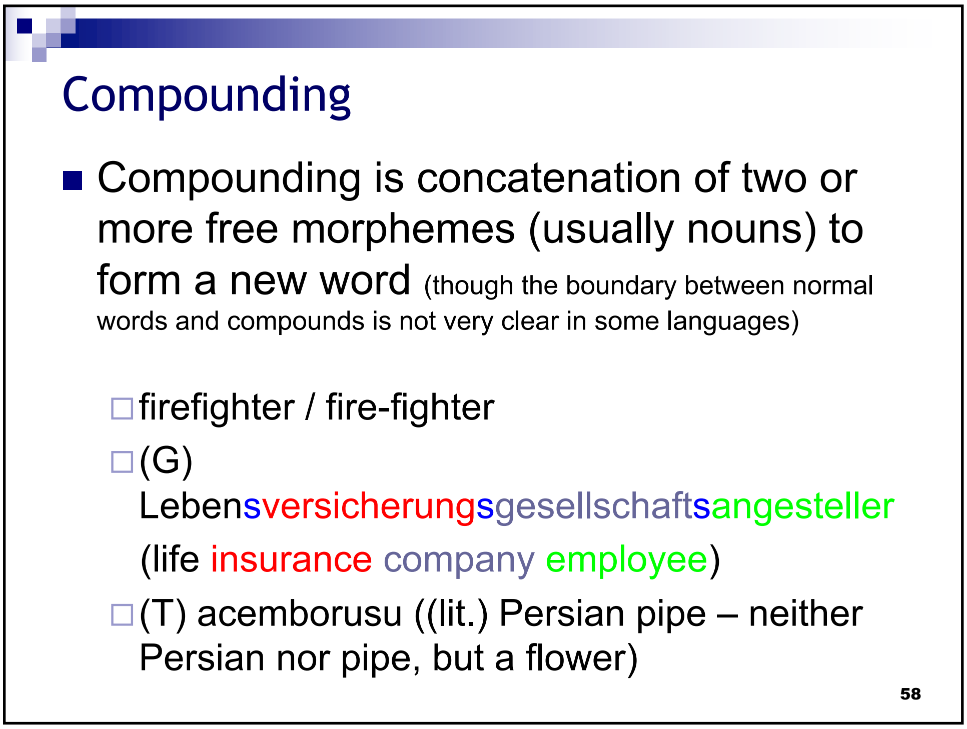
Derivational Morphology

- □happy (Adj) ⇒ happi+ness (Noun)
- □(T) elçi (Noun, ambassador) ⇒
  elçi+lik (Noun, embassy)
- □(G) Botschaft (Noun, embassy) ⇒
  Botschaft+er (Noun, ambassador)
- □(T) git (Verb, go) ⇒
  gid+er+ken (Adverb, while going)



Derivational Morphology

- Productive vs. unproductive derivational morphology
  - □ Productive: can apply to almost all members of a class of words
  - □ Unproductive: applies to only a few members of a class or words
    - lexicalized derivations (e.g., application as in “application program”)



Compounding

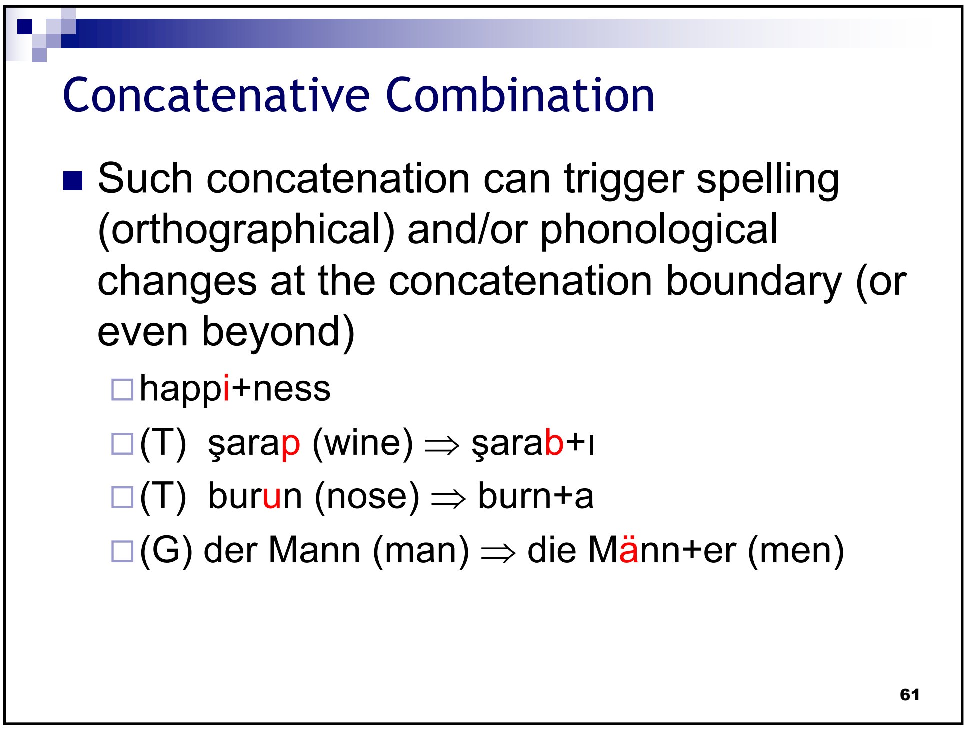
- Compounding is concatenation of two or more free morphemes (usually nouns) to form a new word (though the boundary between normal words and compounds is not very clear in some languages)
  - □ firefighter / fire-fighter
  - □ (G) Lebensversicherungsgesellschaftsangesteller (life insurance company employee)
  - □ (T) acemborusu ((lit.) Persian pipe – neither Persian nor pipe, but a flower)

Combining Morphemes

- Morphemes can be combined in a variety of ways to make up words:
  - □ Concatenative
  - □ Infixation
  - □ Circumfixation
  - □ Templatic Combination
  - □ Reduplication

Concatenative Combination

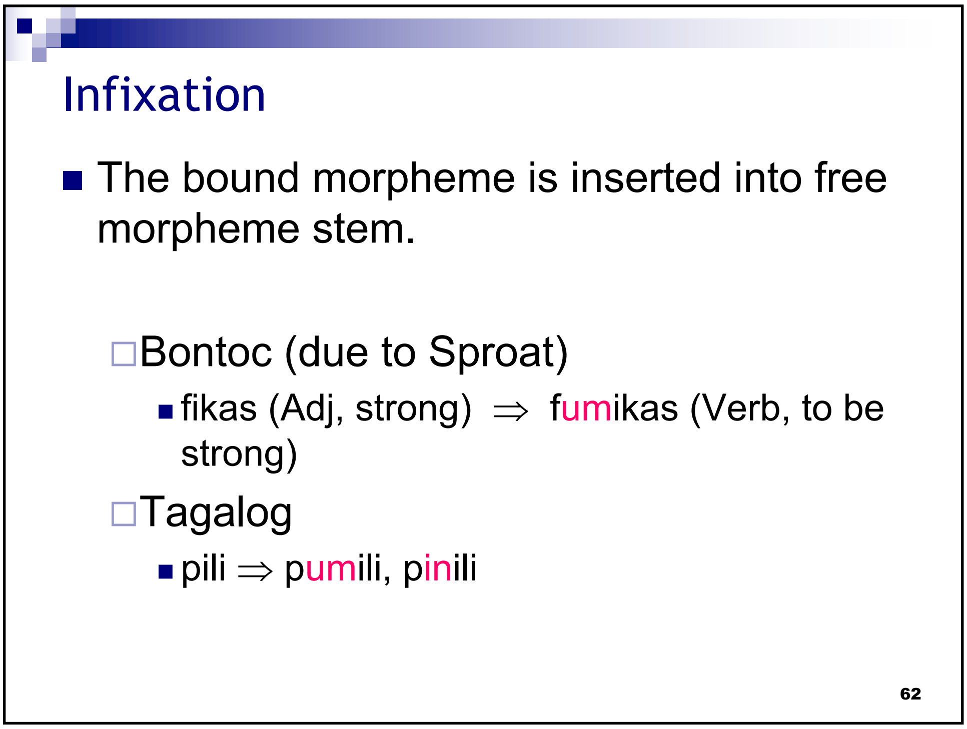
- Bound morphemes are attached before or after the free morpheme (or any other intervening morphemes).
  - □ Prefixation: bound morphemes go before the free morpheme
    - un+happy
  - □ Suffixation: bound morphemes go after the free morpheme
    - happi+ness
      - □ Need to be careful about the order [un+happi]+ness (not un +[happi+ness]
    - el+ler+im+de+ki+ler



Concatenative Combination

- Such concatenation can trigger spelling (orthographical) and/or phonological changes at the concatenation boundary (or even beyond)
  - □happi+ness
  - □(T) şarap (wine) ⇒ şarab+ı
  - □(T) burun (nose) ⇒ burn+a
  - □(G) der Mann (man) ⇒ die Männ+er (men)

61



Infixation

- The bound morpheme is inserted into free morpheme stem.
  - □Bontoc (due to Sproat)
    - fikas (Adj, strong) ⇒ fumikas (Verb, to be strong)
  - □Tagalog
    - pili ⇒ pumili, pinili

62

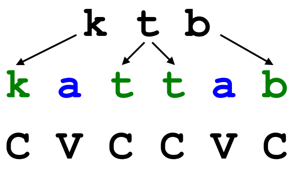
Circumfixation

- Part of the morpheme goes before the stem, part goes after the stem.

- German past participle
  - □ tauschen (Verb, to exchange) ⇒ getauscht (Participle)
  - □ Sagen (Verb, to say) ⇒ gesagt

Templatic Combination

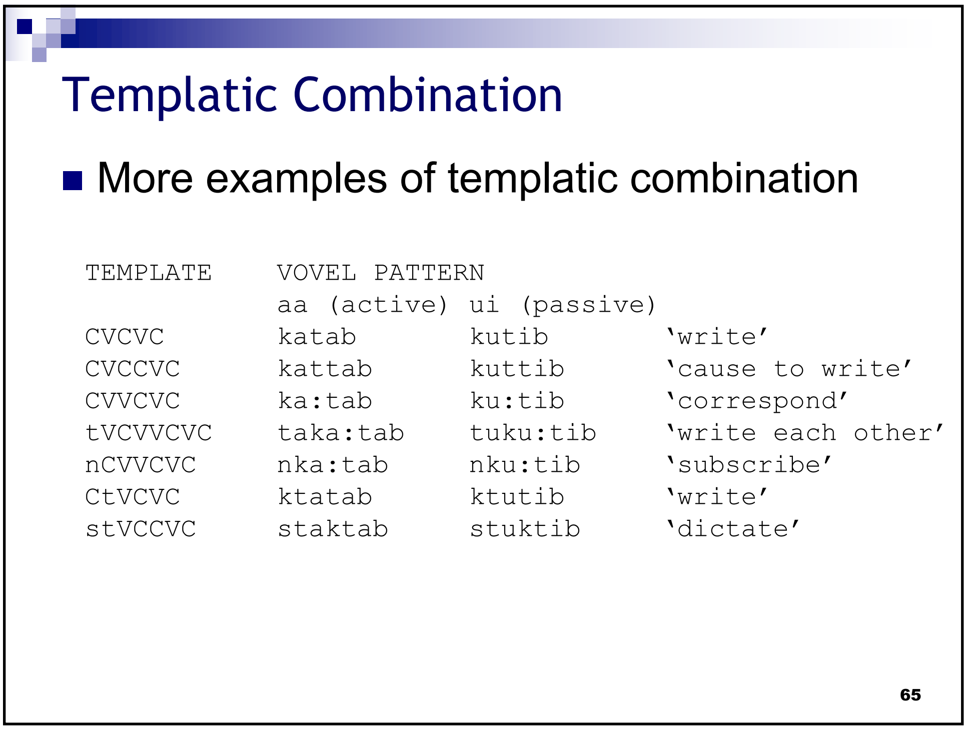
- The root is modulated with a template to generate stem to which other morphemes can be added by concatentaion etc.
- Semitic Languages (e.g., Arabic)
  - □ root ktb (the general concept of writing)
  - □ template CVCCVC
  - □ vocalism (a,a)



k t b

k a t t a b

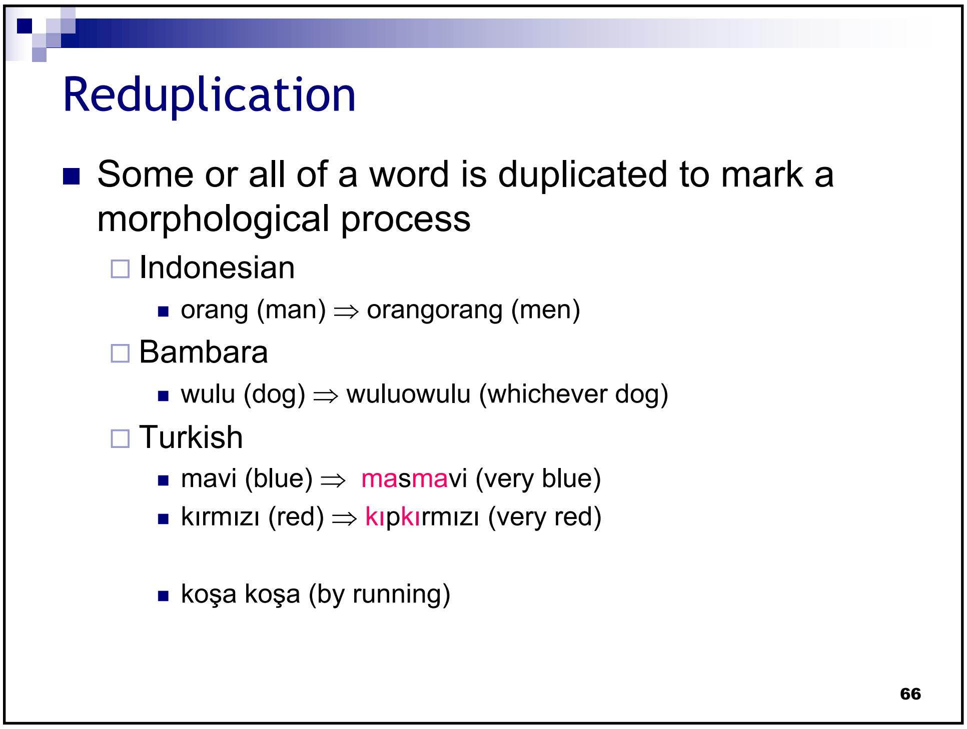
C V C C V C



Templatic Combination

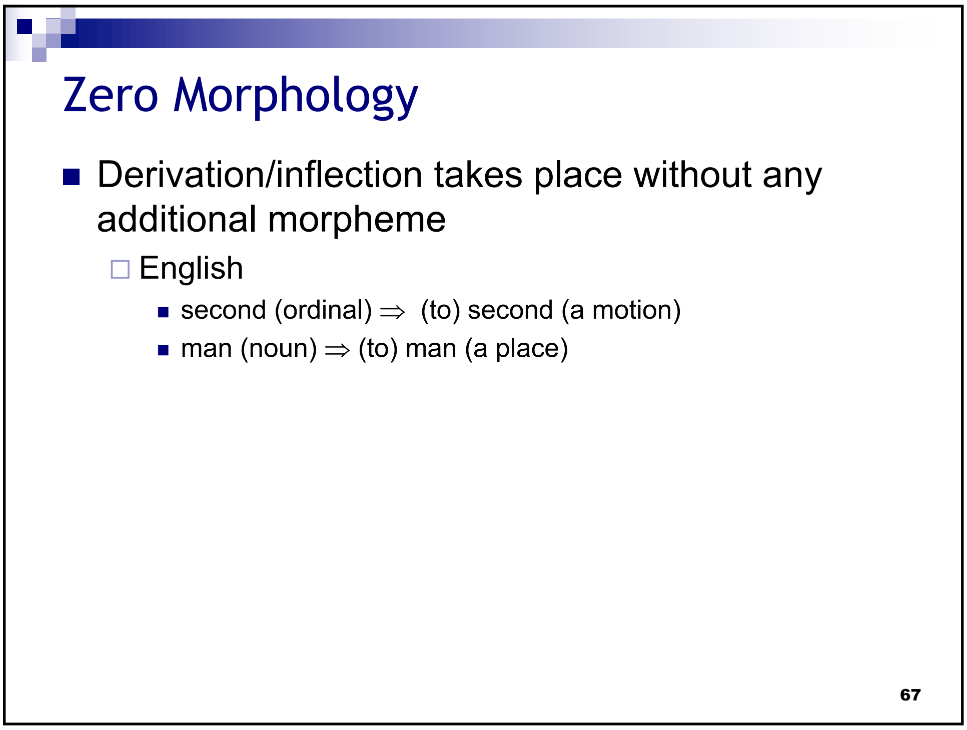
- More examples of templatic combination

TEMPLATE	VOVEL PATTERN		
	aa (active)	ui (passive)	
CVCVC	katab	kutib	'write'
CVCCVC	kattab	kuttib	'cause to write'
CVVCVC	ka:tab	ku:tib	'correspond'
tVCVVCVC	taka:tab	tuku:tib	'write each other'
nCVVCVC	nka:tab	nku:tib	'subscribe'
CtVCVC	ktatab	ktutib	'write'
stVCCVC	staktab	stuktib	'dictate'



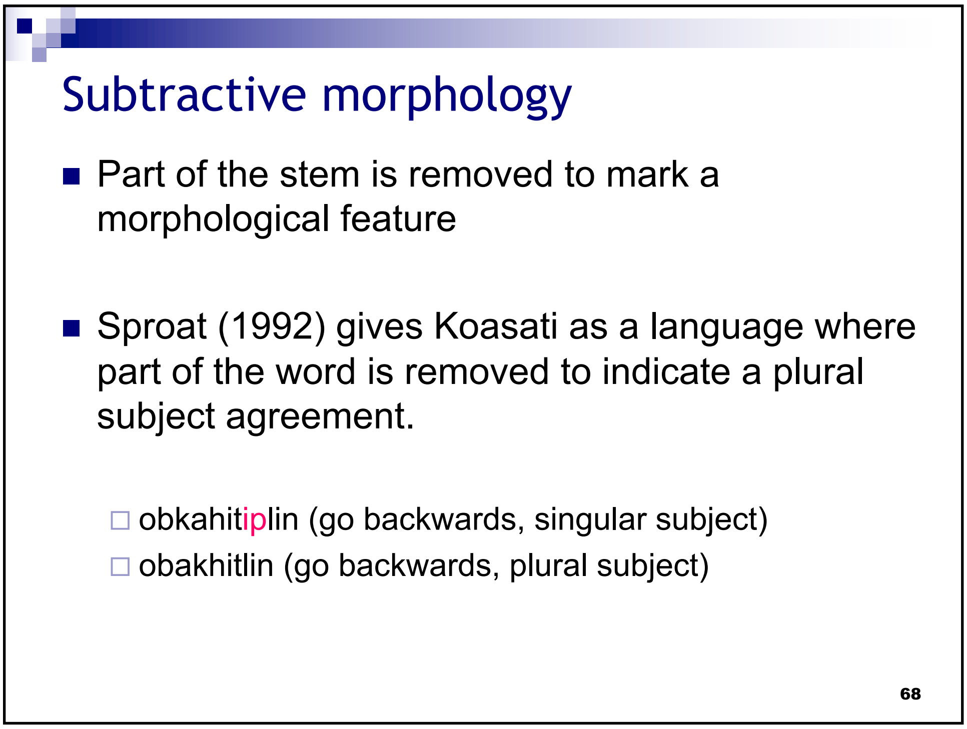
Reduplication

- Some or all of a word is duplicated to mark a morphological process
  - □ Indonesian
    - orang (man) ⇒ orangorang (men)
  - □ Bambara
    - wulu (dog) ⇒ wuluowulu (whichever dog)
  - □ Turkish
    - mavi (blue) ⇒ masmavi (very blue)
    - kırmızı (red) ⇒ kıpkırmızı (very red)
    - koşa koşa (by running)



Zero Morphology

- Derivation/inflection takes place without any additional morpheme
  - □ English
    - second (ordinal) ⇒ (to) second (a motion)
    - man (noun) ⇒ (to) man (a place)



Subtractive morphology

- Part of the stem is removed to mark a morphological feature

- Sproat (1992) gives Koasati as a language where part of the word is removed to indicate a plural subject agreement.

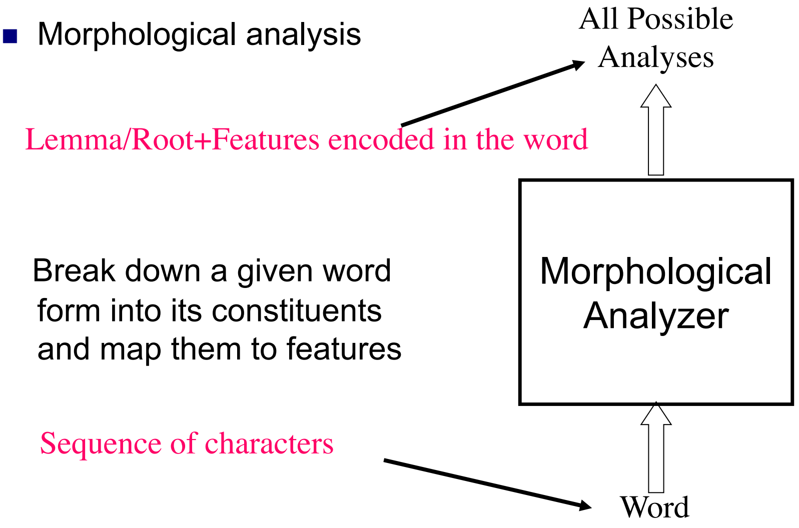
  - □ obkahitiplin (go backwards, singular subject)
  - □ obakhitlin (go backwards, plural subject)

(Back to) Computational Morphology

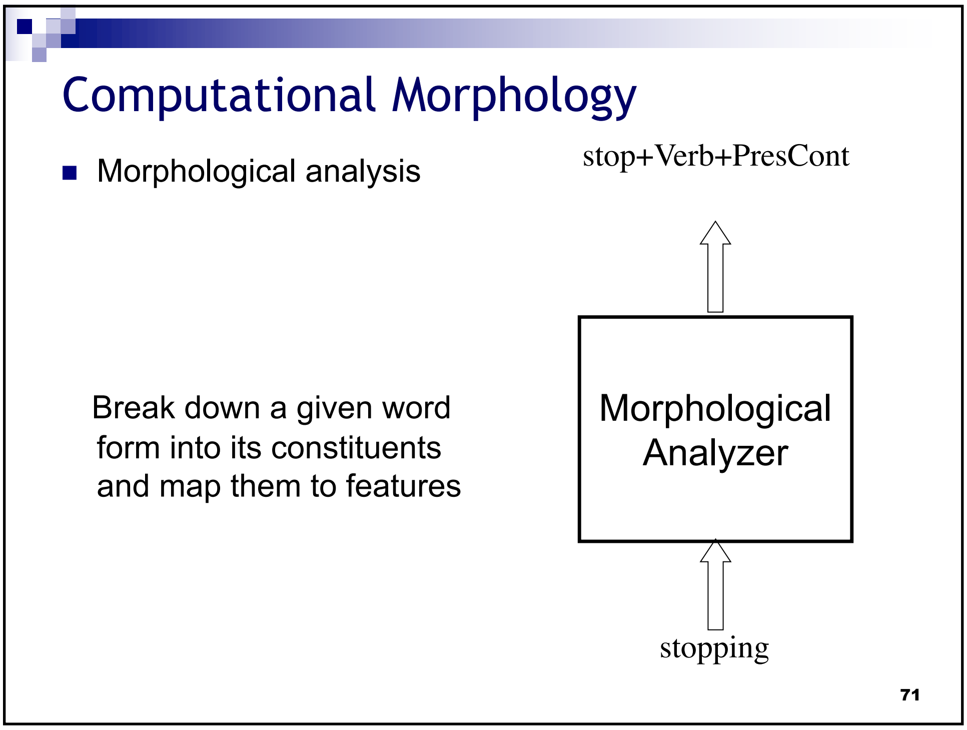
- Computational morphology deals with
  - □ developing theories and techniques for
  - □ computational analysis and synthesis of word forms.
    - Analysis: Separate and identify the constituent morphemes and mark the information they encode
    - Synthesis (Generation): Given a set constituent morphemes or information be encoded, produce the corresponding word(s)

Computational Morphology

- Morphological analysis



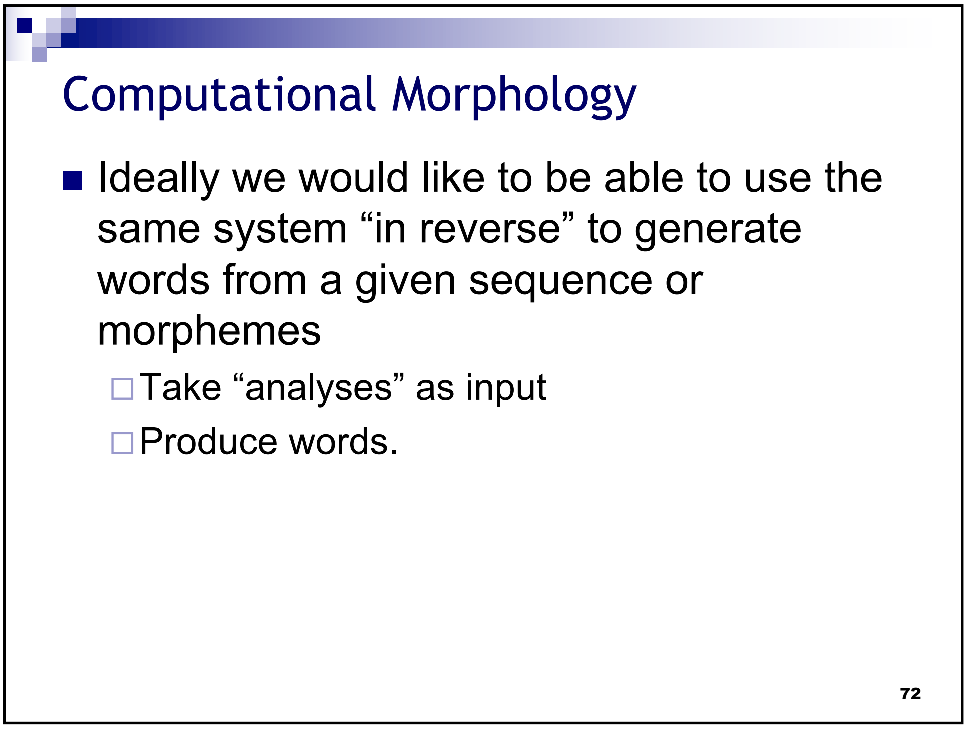
Break down a given word form into its constituents and map them to features



Computational Morphology

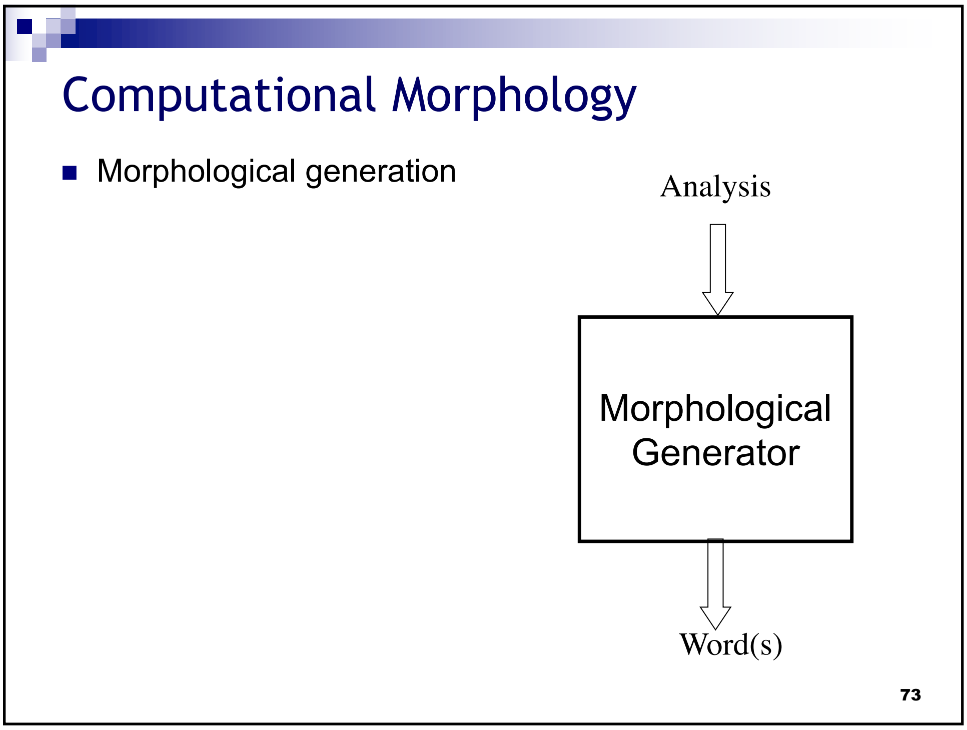
- Morphological analysis

Break down a given word form into its constituents and map them to features



Computational Morphology

- Ideally we would like to be able to use the same system “in reverse” to generate words from a given sequence or morphemes
  - Take “analyses” as input
  - Produce words.



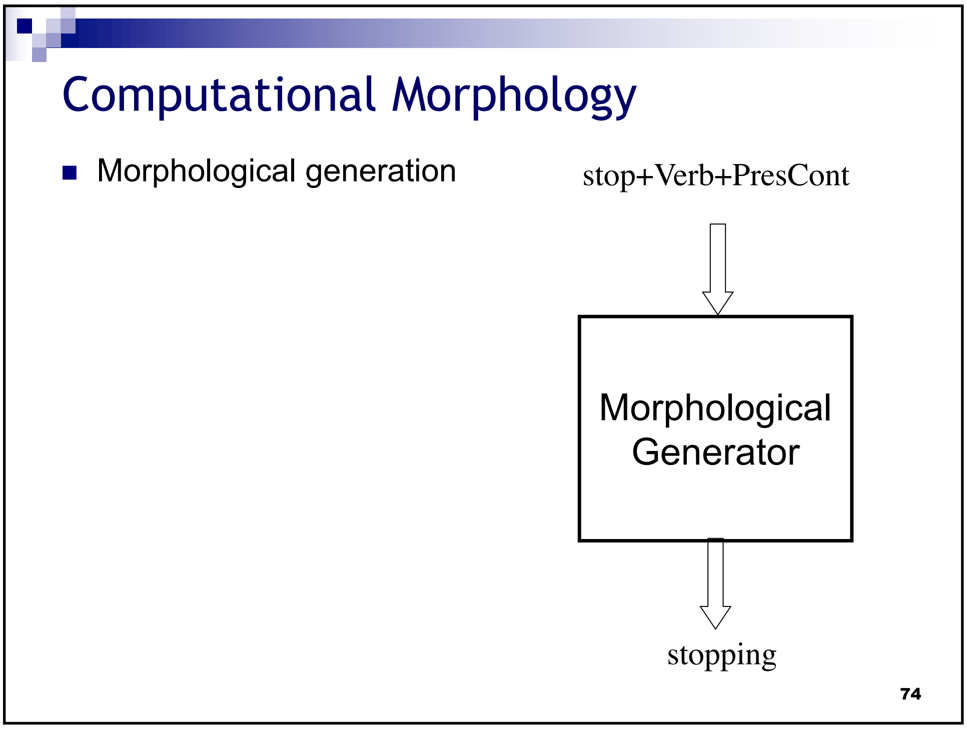
Computational Morphology

- Morphological generation

Analysis

Morphological Generator

Word(s)



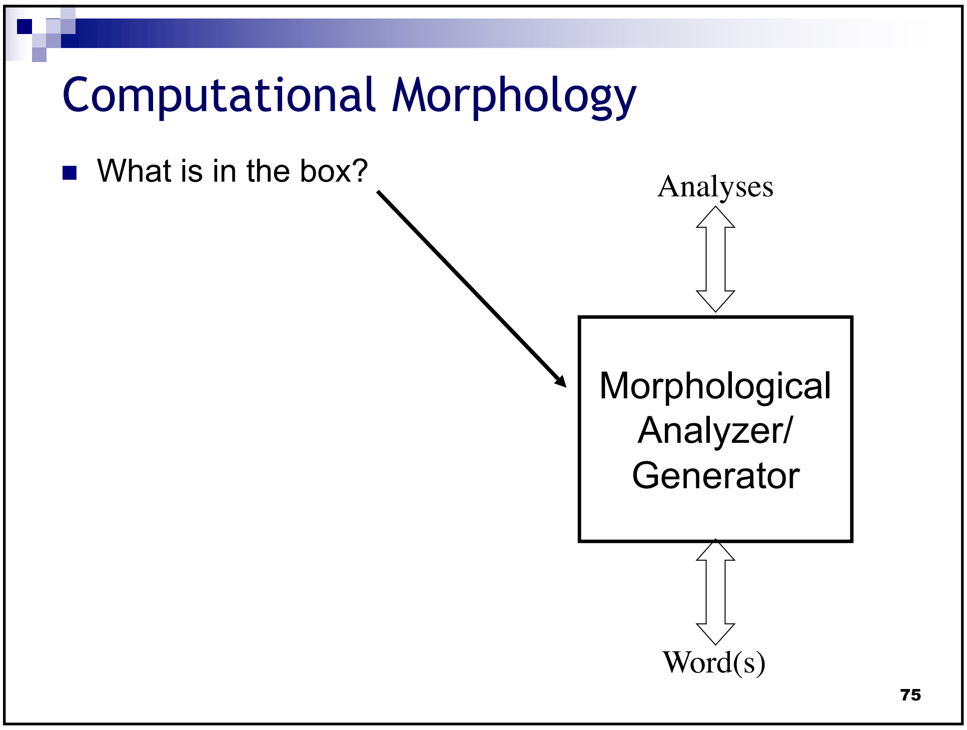
Computational Morphology

- Morphological generation

stop+Verb+PresCont

Morphological Generator

stopping



Computational Morphology

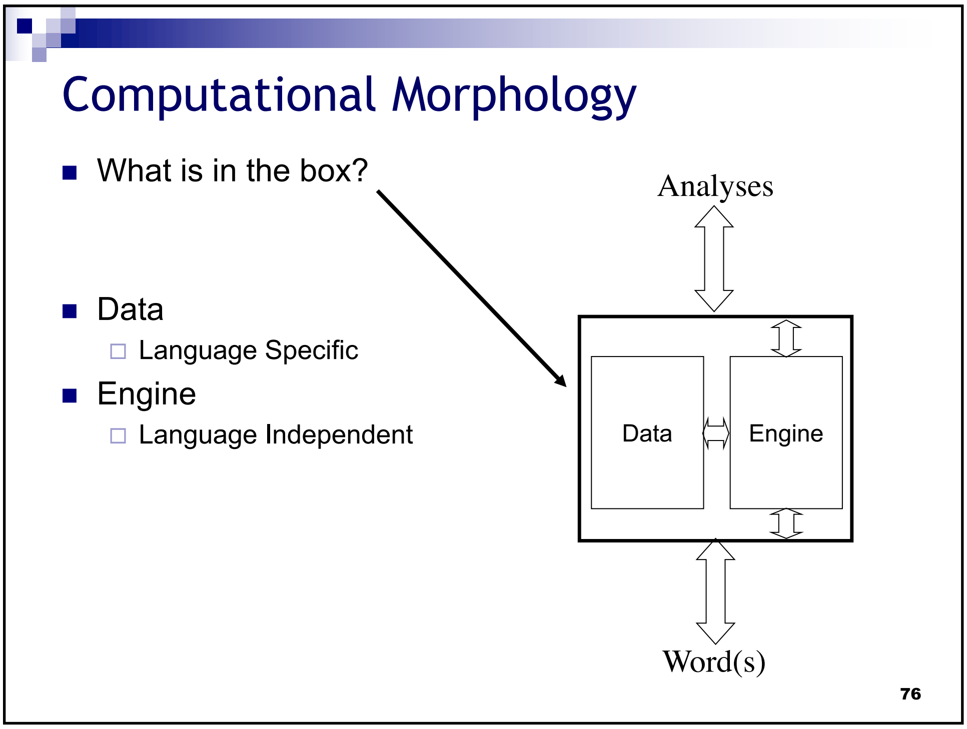
- What is in the box?

Analyses

Morphological Analyzer/ Generator

Word(s)

75



Computational Morphology

- What is in the box?
- Data
  - □ Language Specific
- Engine
  - □ Language Independent

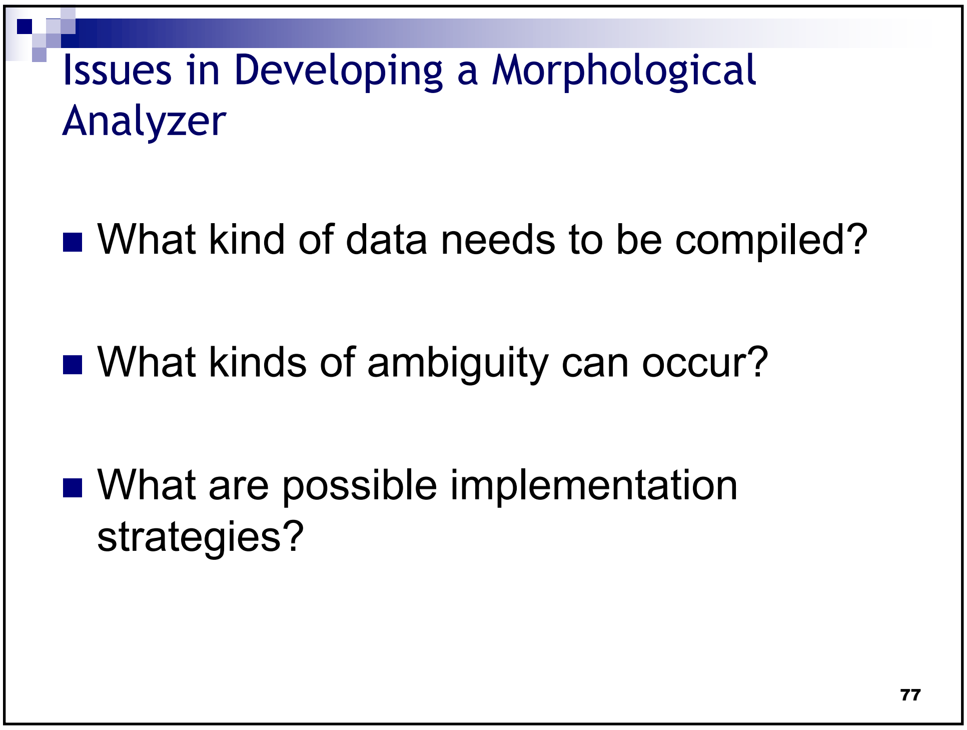
Analyses

Data

Engine

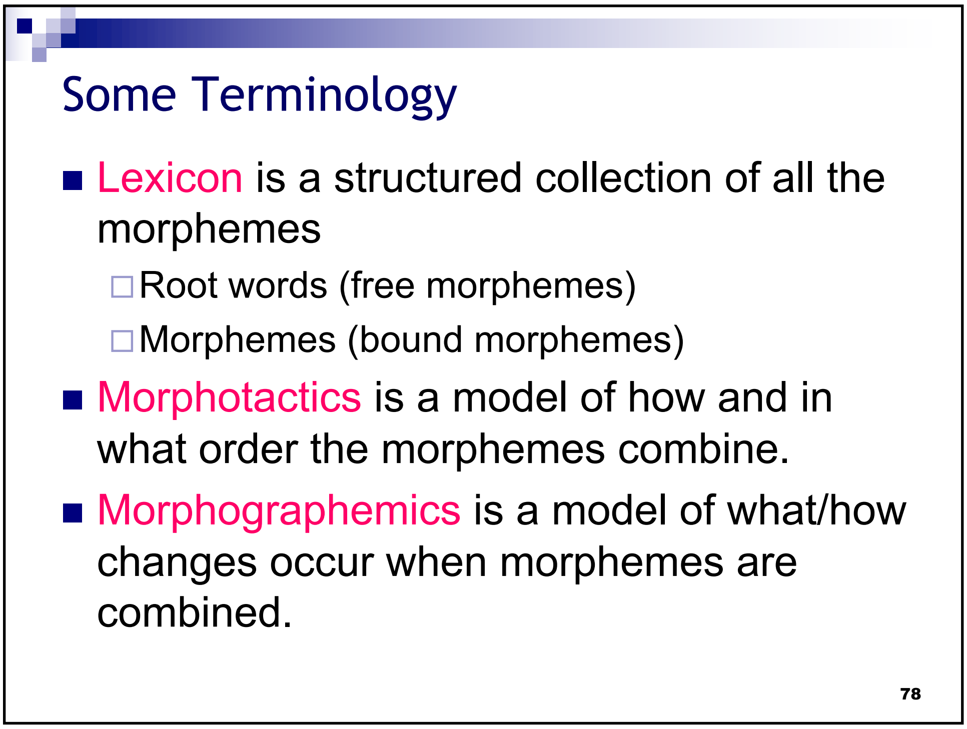
Word(s)

76



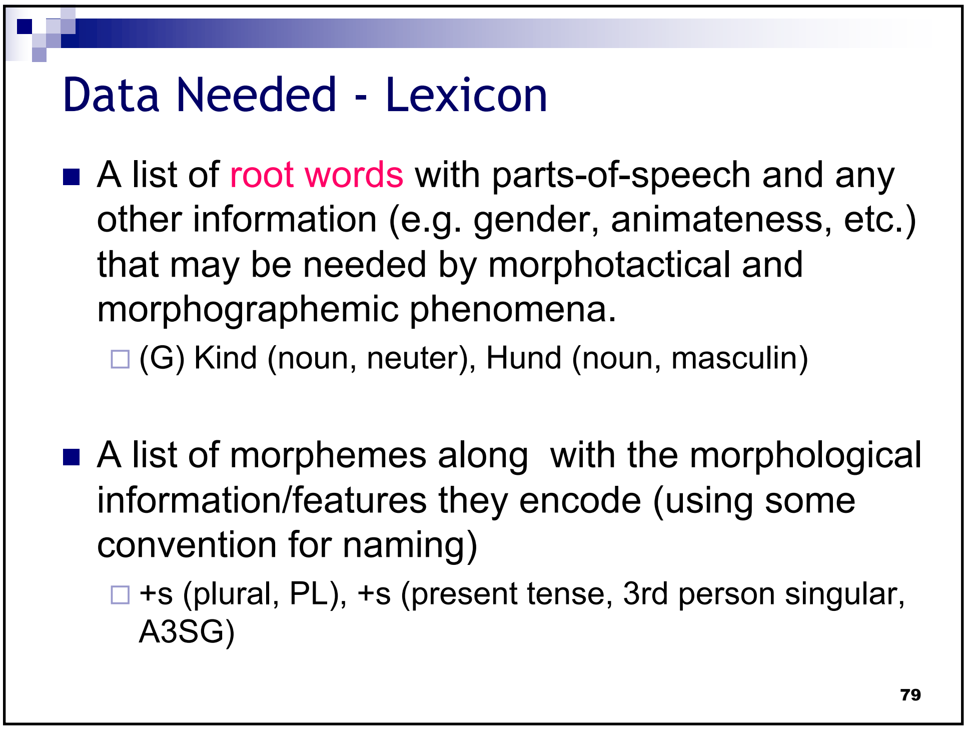
Issues in Developing a Morphological Analyzer

- What kind of data needs to be compiled?
- What kinds of ambiguity can occur?
- What are possible implementation strategies?



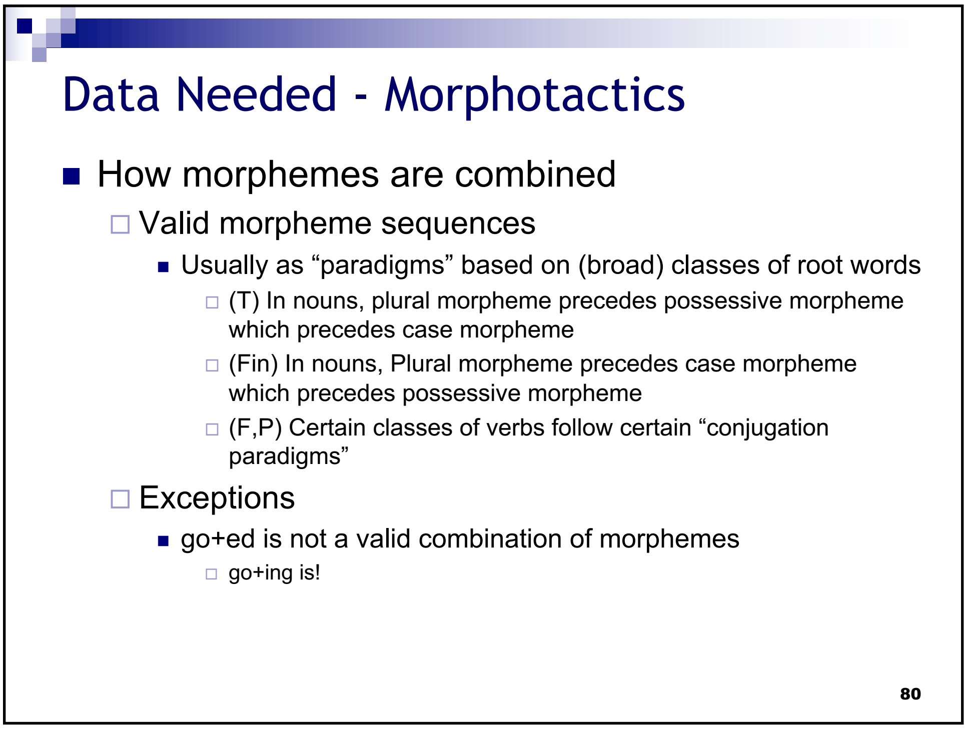
Some Terminology

- Lexicon is a structured collection of all the morphemes
  - □ Root words (free morphemes)
  - □ Morphemes (bound morphemes)
- Morphotactics is a model of how and in what order the morphemes combine.
- Morphographemics is a model of what/how changes occur when morphemes are combined.



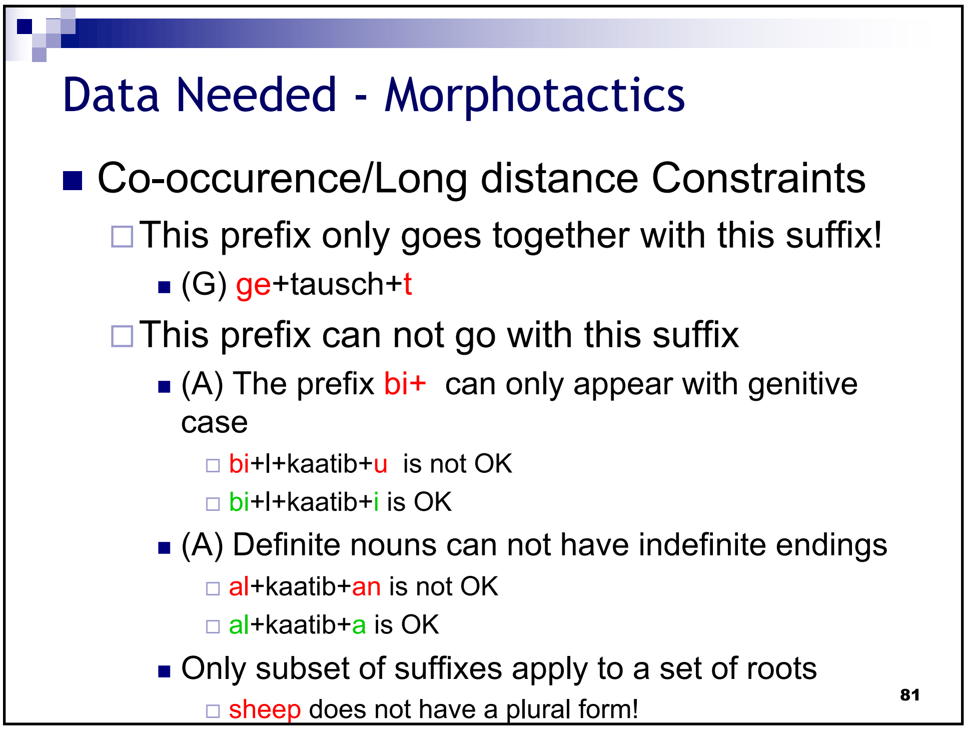
Data Needed - Lexicon

- A list of root words with parts-of-speech and any other information (e.g. gender, animateness, etc.) that may be needed by morphotactical and morphographemic phenomena.
  - (G) Kind (noun, neuter), Hund (noun, masculin)
- A list of morphemes along with the morphological information/features they encode (using some convention for naming)
  - +s (plural, PL), +s (present tense, 3rd person singular, A3SG)



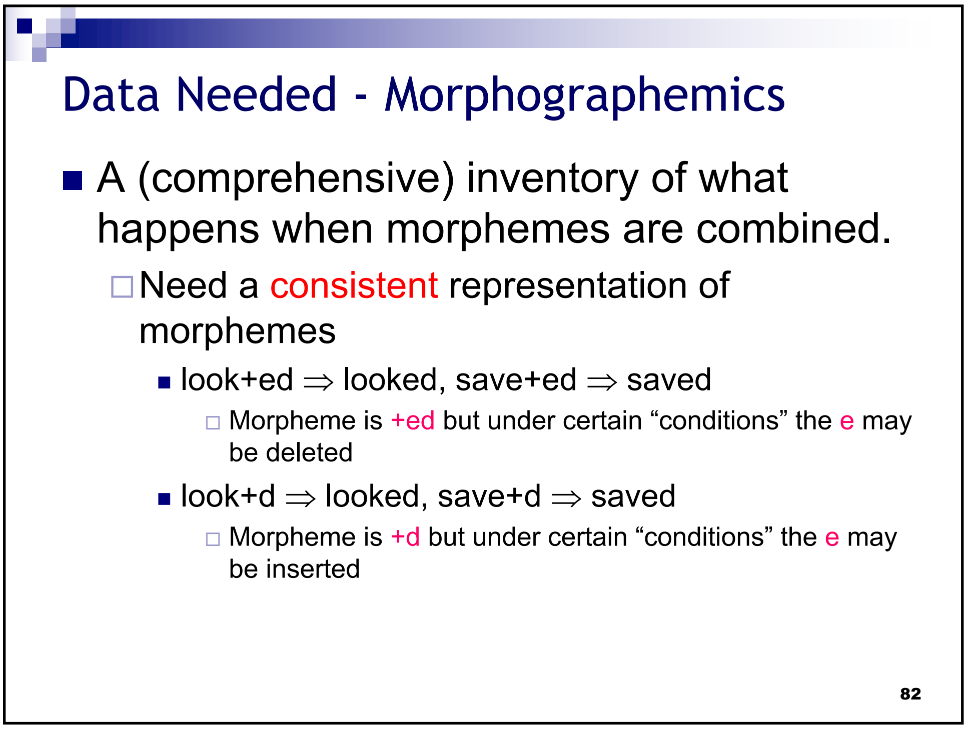
Data Needed - Morphotactics

- How morphemes are combined
  - Valid morpheme sequences
    - Usually as “paradigms” based on (broad) classes of root words
      - (T) In nouns, plural morpheme precedes possessive morpheme which precedes case morpheme
      - (Fin) In nouns, Plural morpheme precedes case morpheme which precedes possessive morpheme
      - (F,P) Certain classes of verbs follow certain “conjugation paradigms”
  - Exceptions
    - go+ed is not a valid combination of morphemes
      - go+ing is!



Data Needed - Morphotactics

- Co-occurence/Long distance Constraints
  - This prefix only goes together with this suffix!
    - (G) ge+tausch+t
  - This prefix can not go with this suffix
    - (A) The prefix bi+ can only appear with genitive case
      - bi+l+kaatib+u is not OK
      - bi+l+kaatib+i is OK
    - (A) Definite nouns can not have indefinite endings
      - al+kaatib+an is not OK
      - al+kaatib+a is OK
    - Only subset of suffixes apply to a set of roots
      - sheep does not have a plural form!



Data Needed - Morphographemics

- A (comprehensive) inventory of what happens when morphemes are combined.
  - Need a consistent representation of morphemes
    - look+ed ⇒ looked, save+ed ⇒ saved
      - Morpheme is +ed but under certain “conditions” the e may be deleted
    - look+d ⇒ looked, save+d ⇒ saved
      - Morpheme is +d but under certain “conditions” the e may be inserted

Representation

- Lexical form: An underlying representation of morphemes w/o any morphographemic changes applied.
  - □ easy+est
  - □ shop+ed
  - □ blemish+es
  - □ vase+es
- Surface Form: The actual written form
  - □ easiest
  - □ shopped
  - □ blemishes
  - □ vases

83

Representation

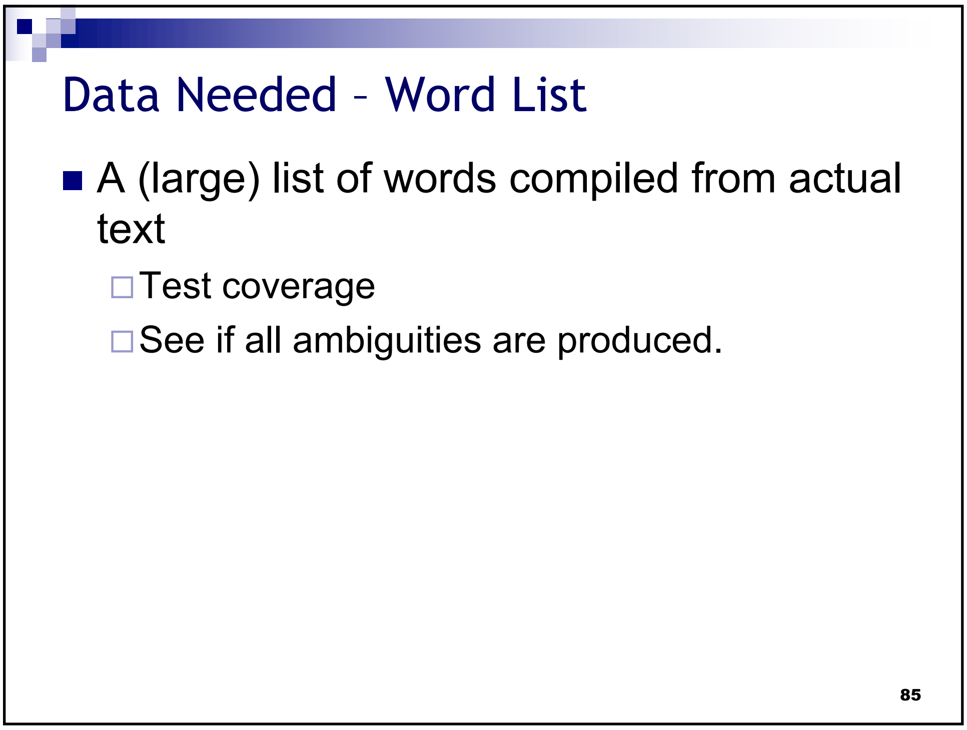
- Lexical form: An underlying representation of morphemes w/o any morphographemic changes applied.
  - □ ev+lAr
  - □ oda+lAr
  - □ tarak+sH
  - □ kese+sH

A={a,e} ← Abstract meta-phonemes

H={ı, i, u, ü}

- Surface Form: The actual written form
  - □ evler
  - □ odalar
  - □ tarağı
  - □ kesesi

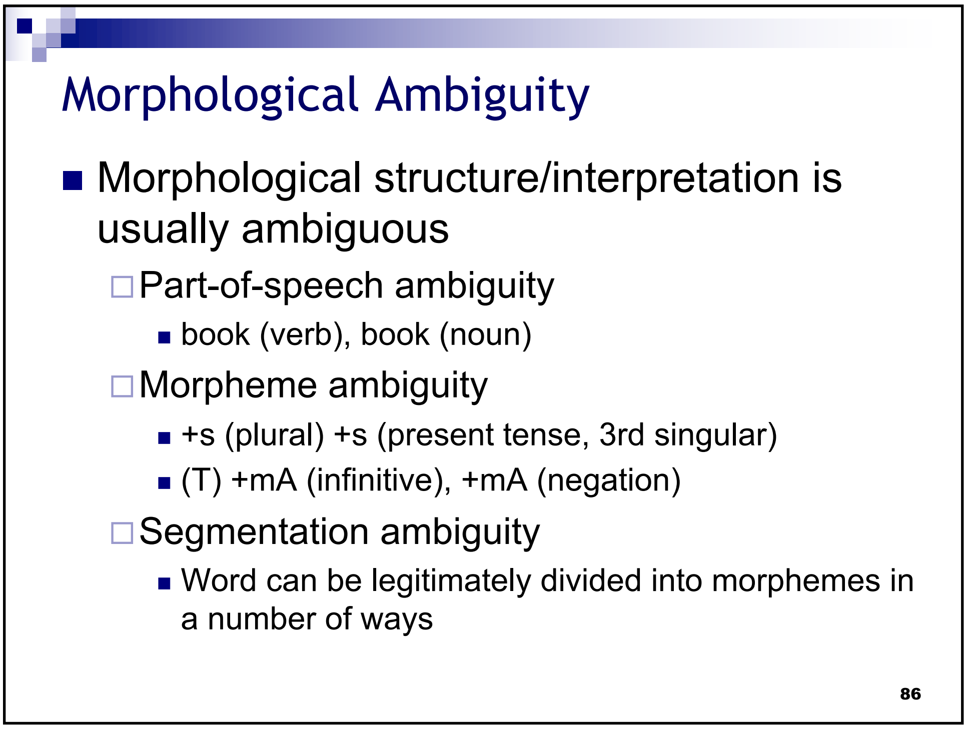
84



Data Needed – Word List

- A (large) list of words compiled from actual text
  - □ Test coverage
  - □ See if all ambiguities are produced.

85



Morphological Ambiguity

- Morphological structure/interpretation is usually ambiguous
  - □ Part-of-speech ambiguity
    - book (verb), book (noun)
  - □ Morpheme ambiguity
    - +s (plural) +s (present tense, 3rd singular)
    - (T) +mA (infinitive), +mA (negation)
  - □ Segmentation ambiguity
    - Word can be legitimately divided into morphemes in a number of ways

86

Morphological Ambiguity

- The same surface form is interpreted in many possible ways in different syntactic contexts.

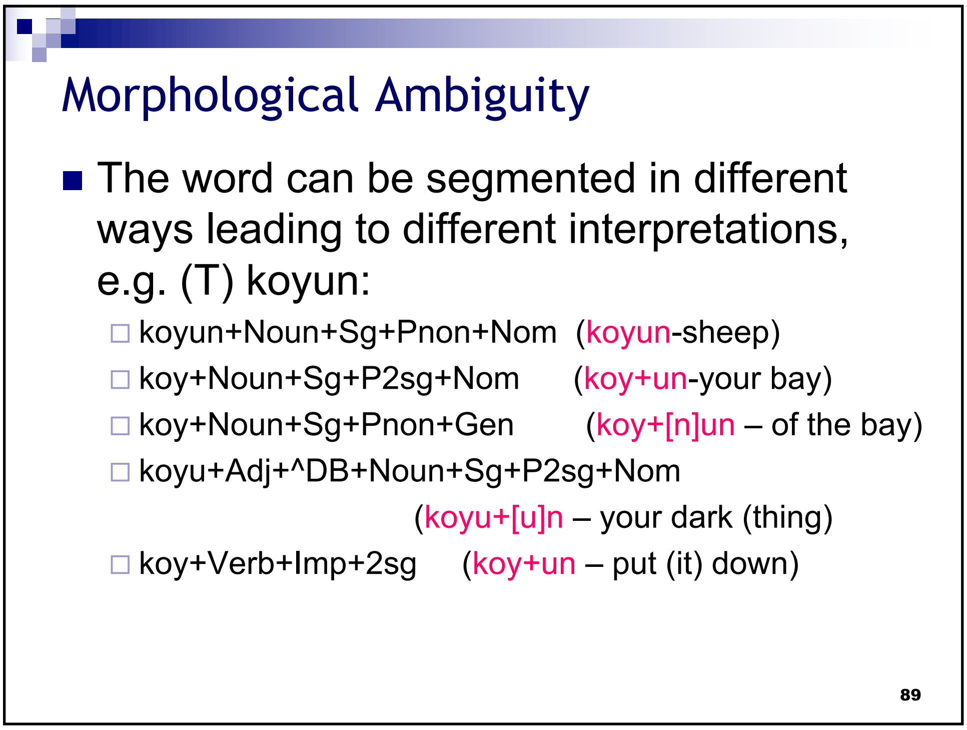
(F) danse
danse+Verb+Subj+3sg (lest s/he dance)
danse+Verb+Subj+1sg (lest I dance)
danse+Verb+Imp+2sg ((you) dance!)
danse+Verb+Ind+3sg ((s/he) dances)
danse+Verb+Ind+1sg ((I) dance)
danse+Noun+Fem+Sg (dance)

(E) read
read+Verb+Pres+N3sg (VBP-I/you/we/they read)
read+Verb+Past (VBD - read past tense)
read+Verb+Participle+(VBN – participle form)
read+Verb (VB - infinitive form)
read+Noun+Sg (NN – singular noun)

Morphological Ambiguity

- The same morpheme can be interpreted differently depending on its position in the morpheme order:

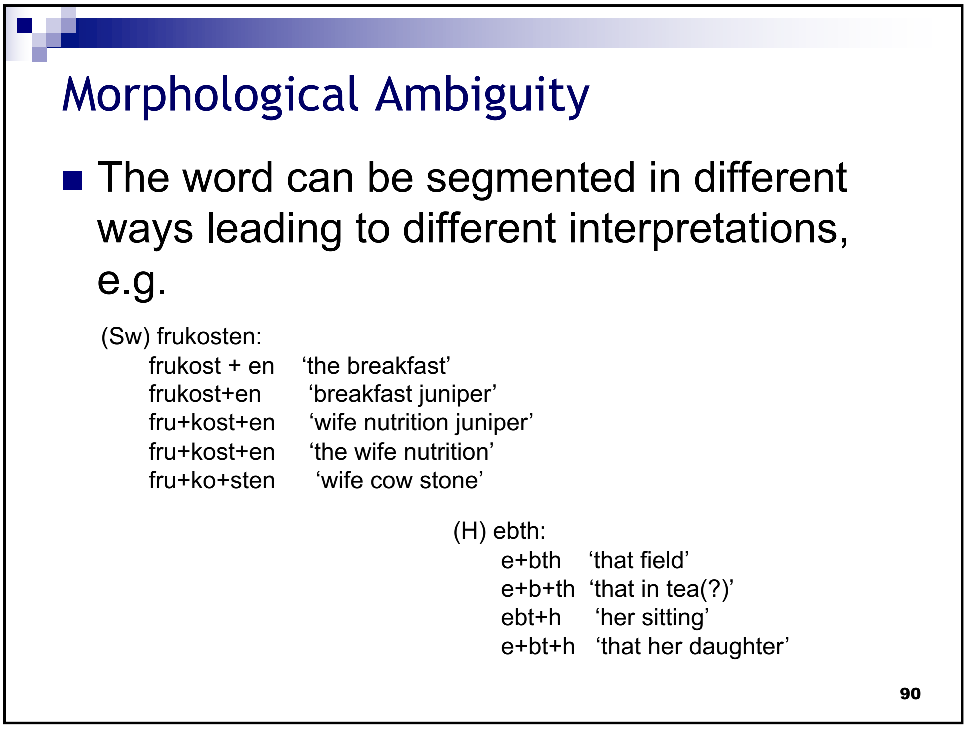
□ (T) git+me ((the act of) going),
□ git+me (don’t go)



Morphological Ambiguity

- The word can be segmented in different ways leading to different interpretations, e.g. (T) koyun:
  - □ koyun+Noun+Sg+Pnon+Nom (koyun-sheep)
  - □ koy+Noun+Sg+P2sg+Nom (koy+un-your bay)
  - □ koy+Noun+Sg+Pnon+Gen (koy+[n]un – of the bay)
  - □ koyu+Adj+^DB+Noun+Sg+P2sg+Nom (koyu+[u]n – your dark (thing)
  - □ koy+Verb+Imp+2sg (koy+un – put (it) down)

89



Morphological Ambiguity

- The word can be segmented in different ways leading to different interpretations, e.g.

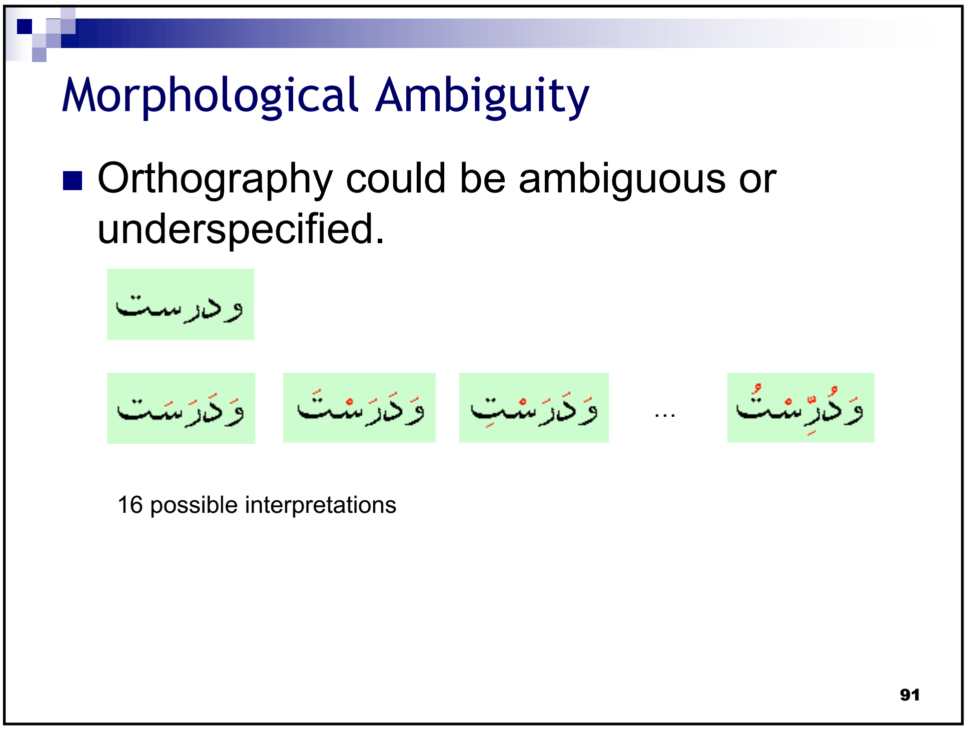
(Sw) frukosten:

frukost + en	‘the breakfast’
frukost+en	‘breakfast juniper’
fru+kost+en	‘wife nutrition juniper’
fru+kost+en	‘the wife nutrition’
fru+ko+sten	‘wife cow stone’

(H) ebth:

e+bth	‘that field’
e+b+th	‘that in tea(?)’
ebt+h	‘her sitting’
e+bt+h	‘that her daughter’

90



Morphological Ambiguity

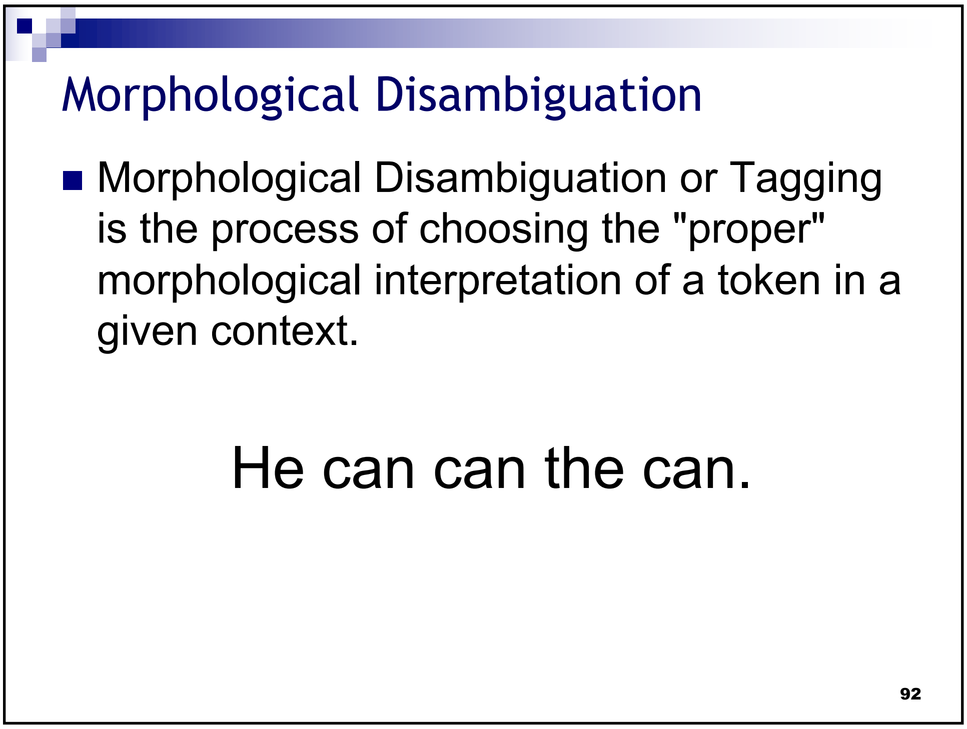
- Orthography could be ambiguous or underspecified.

ودرست

وَدَرَسَت   وَدَرَسْتَ   وَدَرَسْتِ   …   وَدُرِّسْتُ

16 possible interpretations

91

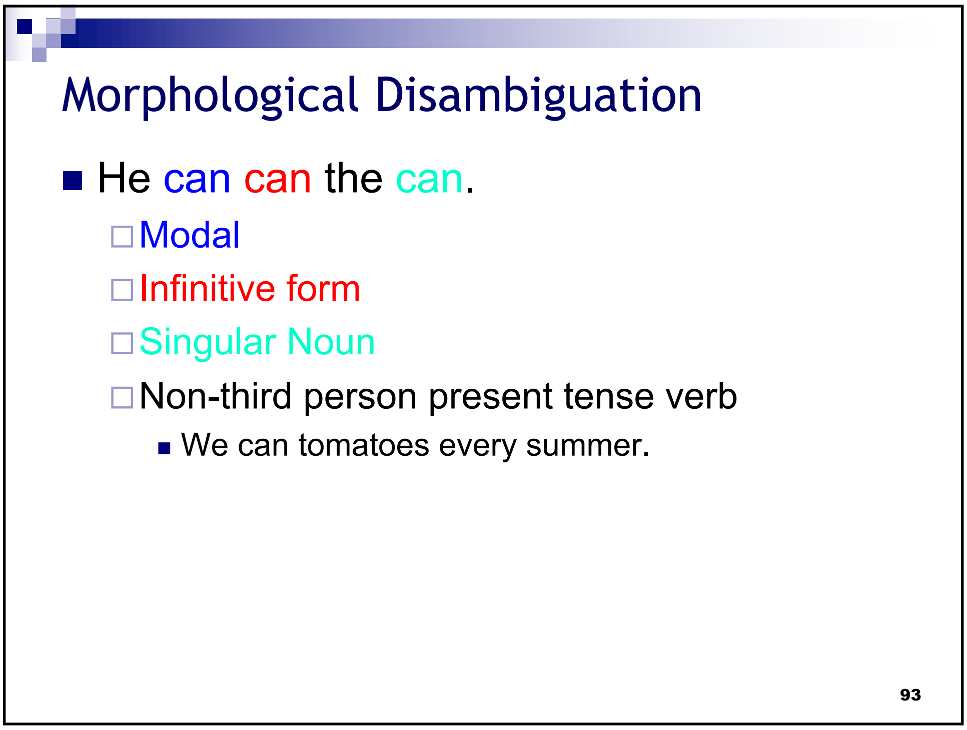


Morphological Disambiguation

- Morphological Disambiguation or Tagging is the process of choosing the "proper" morphological interpretation of a token in a given context.

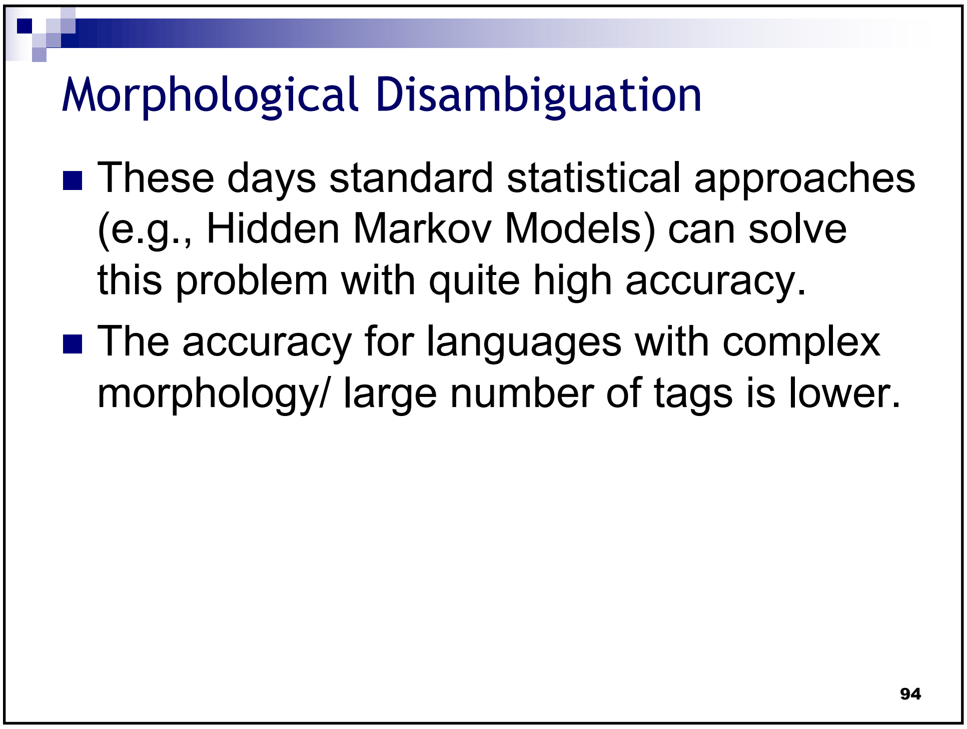
He can can the can.

92



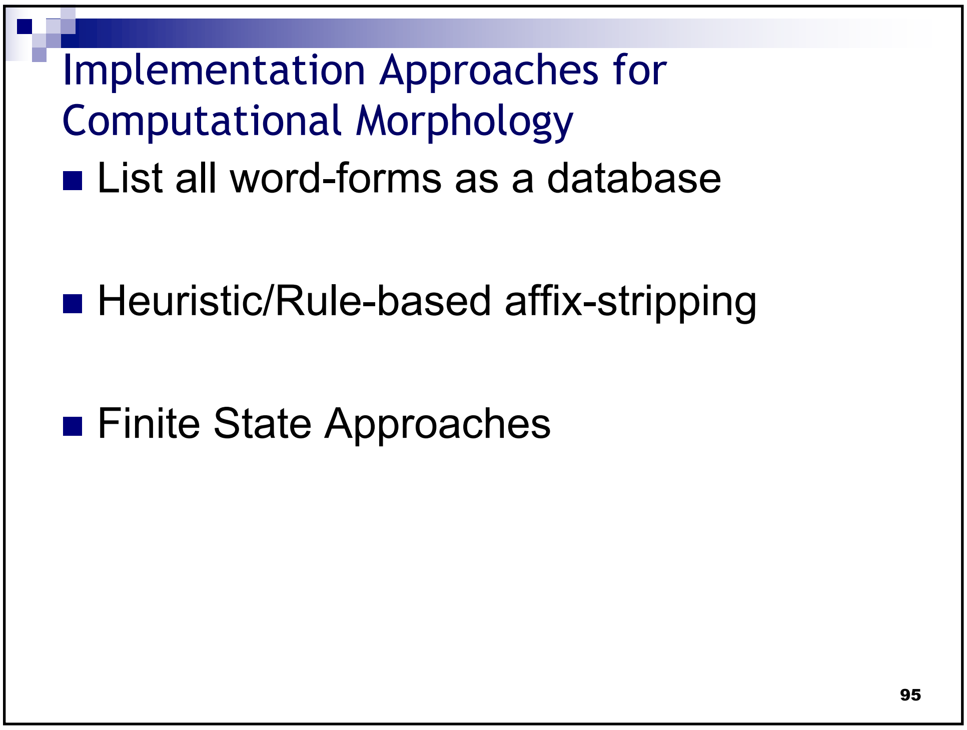
Morphological Disambiguation

- He can can the can.
  - □ Modal
  - □ Infinitive form
  - □ Singular Noun
  - □ Non-third person present tense verb
    - We can tomatoes every summer.



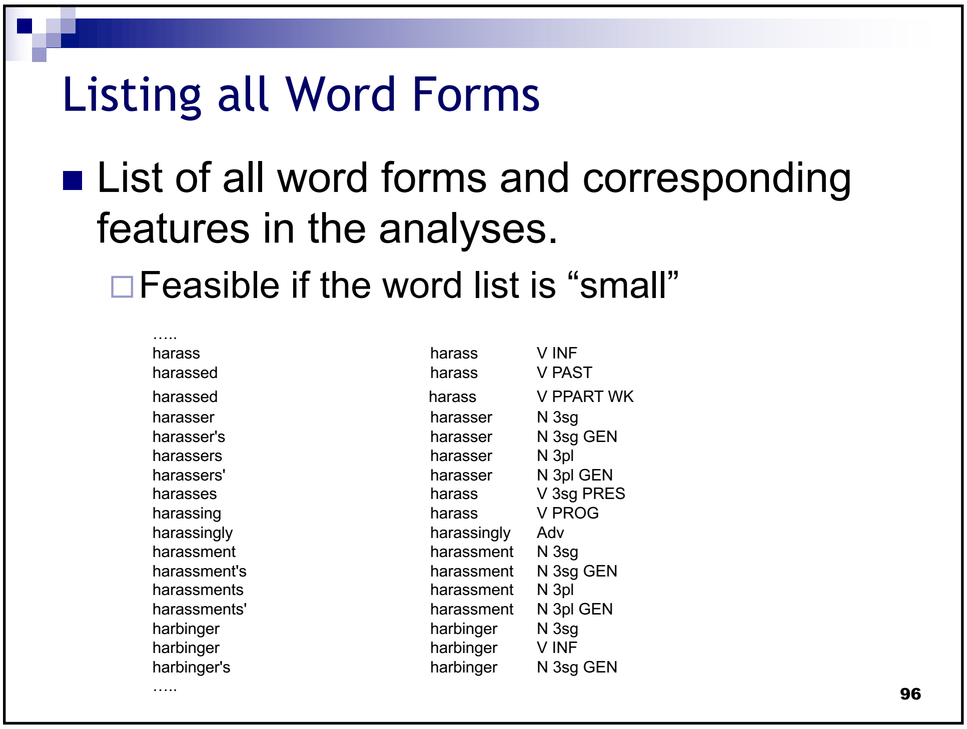
Morphological Disambiguation

- These days standard statistical approaches (e.g., Hidden Markov Models) can solve this problem with quite high accuracy.
- The accuracy for languages with complex morphology/ large number of tags is lower.



Implementation Approaches for Computational Morphology

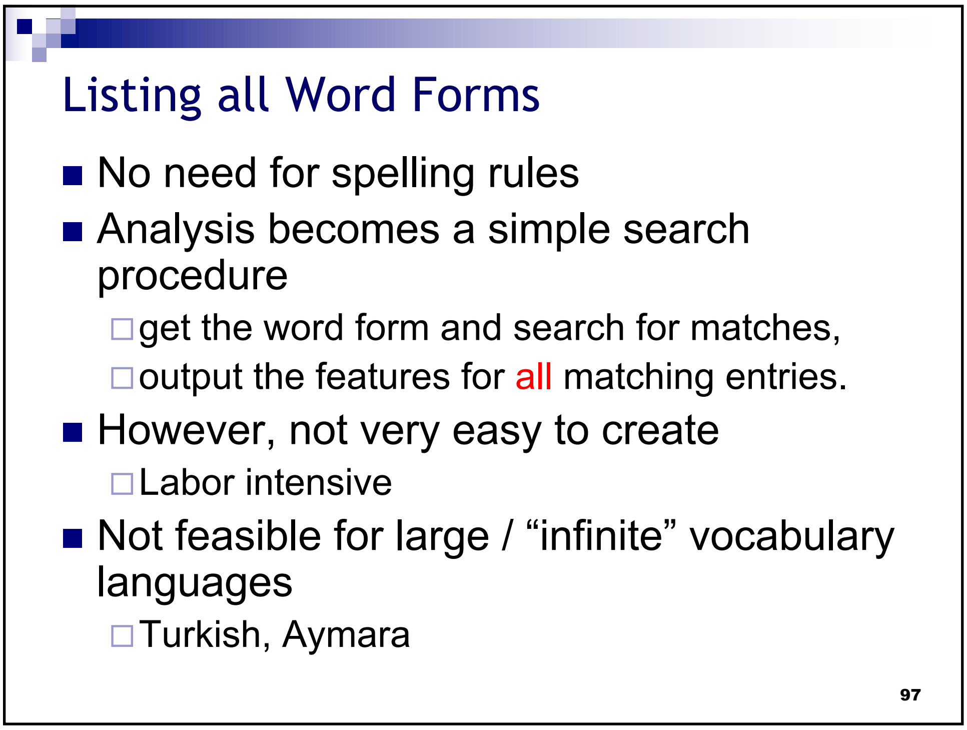
- List all word-forms as a database
- Heuristic/Rule-based affix-stripping
- Finite State Approaches



Listing all Word Forms

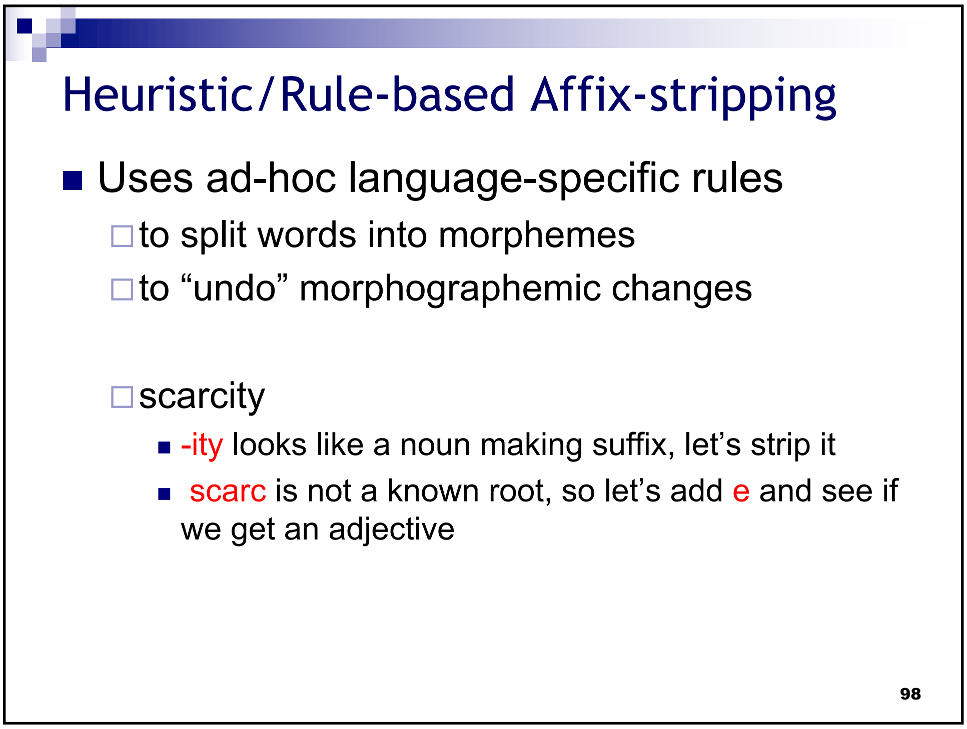
- List of all word forms and corresponding features in the analyses.
  - □ Feasible if the word list is “small”

…..		
harass	harass	V INF
harassed	harass	V PAST
harassed	harass	V PPART WK
harasser	harasser	N 3sg
harasser's	harasser	N 3sg GEN
harassers	harasser	N 3pl
harassers'	harasser	N 3pl GEN
harasses	harass	V 3sg PRES
harassing	harass	V PROG
harassingly	harassingly	Adv
harassment	harassment	N 3sg
harassment's	harassment	N 3sg GEN
harassments	harassment	N 3pl
harassments'	harassment	N 3pl GEN
harbinger	harbinger	N 3sg
harbinger	harbinger	V INF
harbinger's	harbinger	N 3sg GEN
…..		



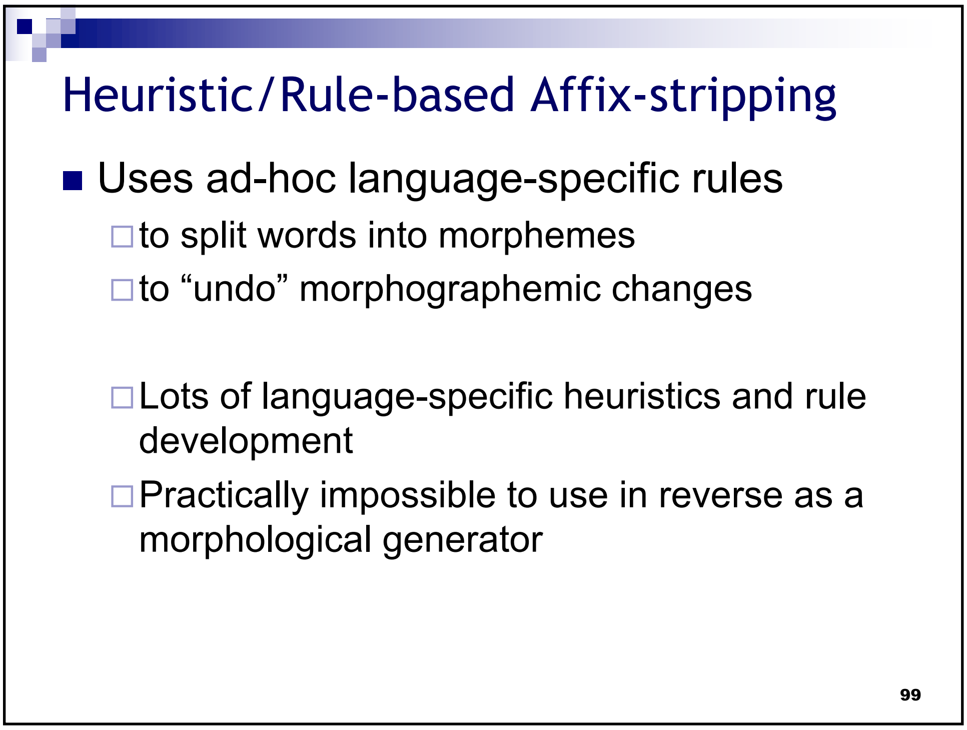
Listing all Word Forms

- No need for spelling rules
- Analysis becomes a simple search procedure
  - □ get the word form and search for matches,
  - □ output the features for all matching entries.
- However, not very easy to create
  - □ Labor intensive
- Not feasible for large / “infinite” vocabulary languages
  - □ Turkish, Aymara



Heuristic/Rule-based Affix-stripping

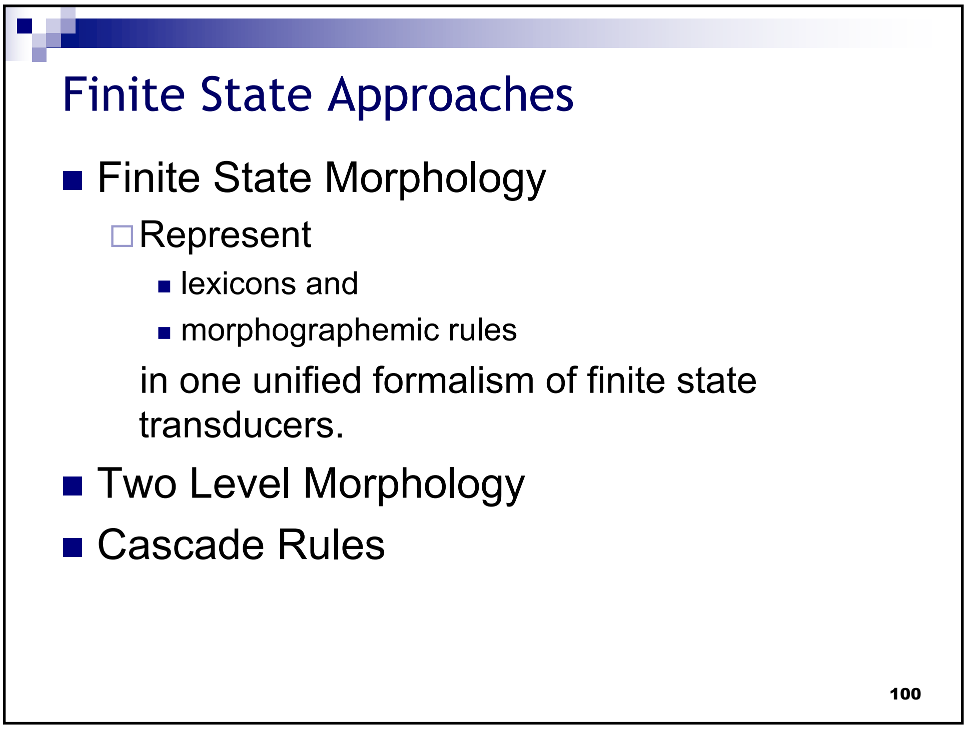
- Uses ad-hoc language-specific rules
  - □ to split words into morphemes
  - □ to “undo” morphographemic changes
  - □ scarcity
    - -ity looks like a noun making suffix, let’s strip it
    - scarc is not a known root, so let’s add e and see if we get an adjective



Heuristic/Rule-based Affix-stripping

- Uses ad-hoc language-specific rules
  - □ to split words into morphemes
  - □ to “undo” morphographemic changes

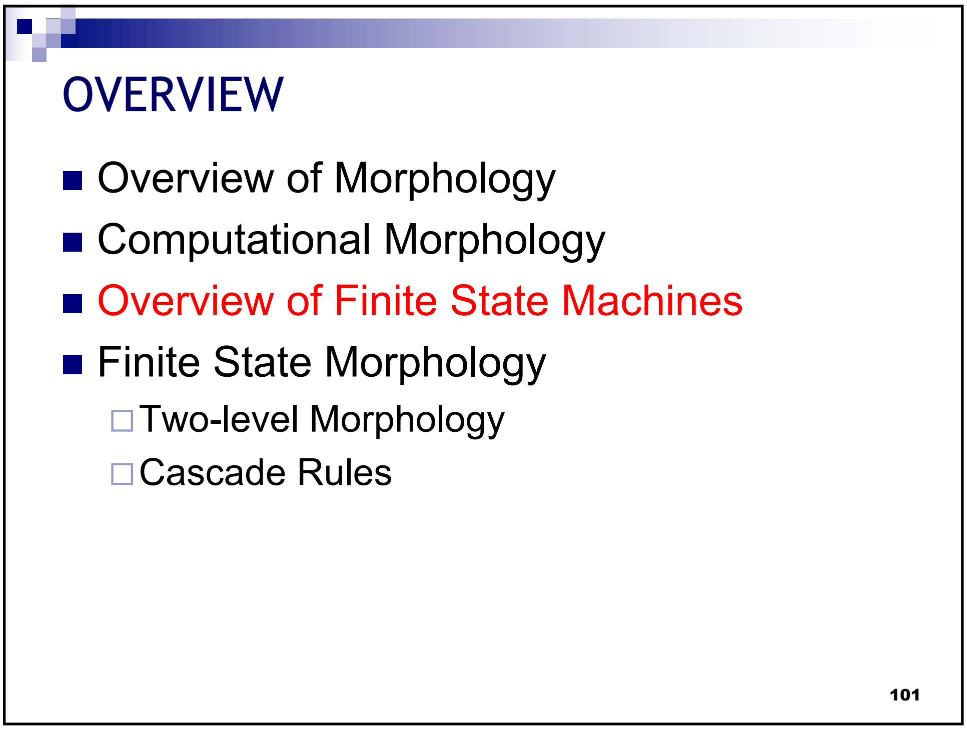
  - □ Lots of language-specific heuristics and rule development
  - □ Practically impossible to use in reverse as a morphological generator



Finite State Approaches

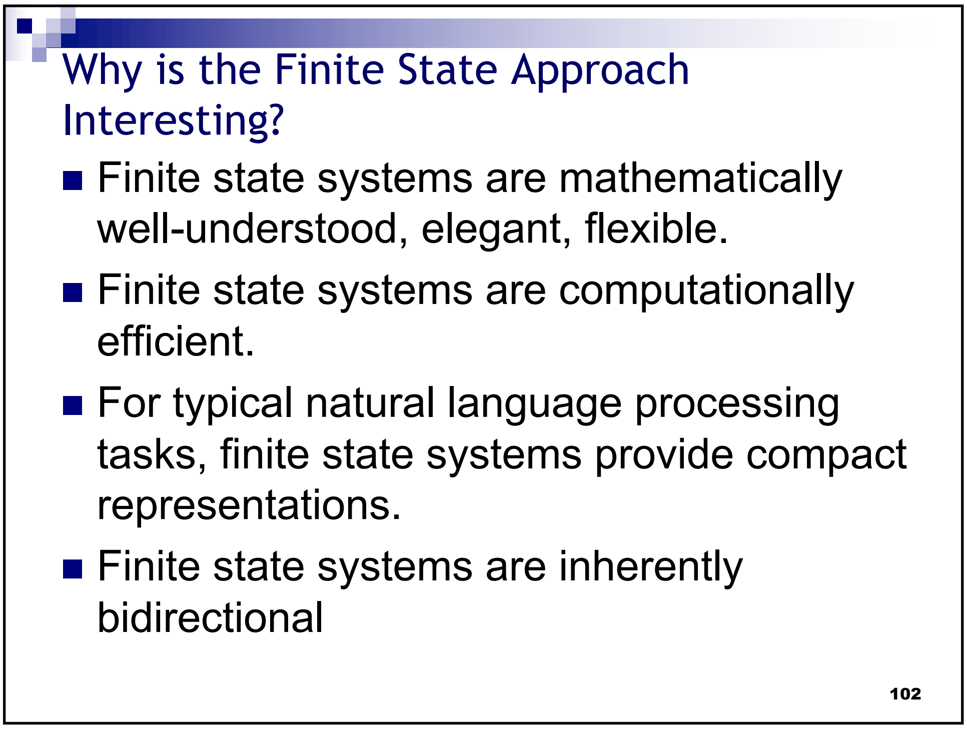
- Finite State Morphology
  - □ Represent
    - lexicons and
    - morphographemic rules

    in one unified formalism of finite state transducers.
- Two Level Morphology
- Cascade Rules



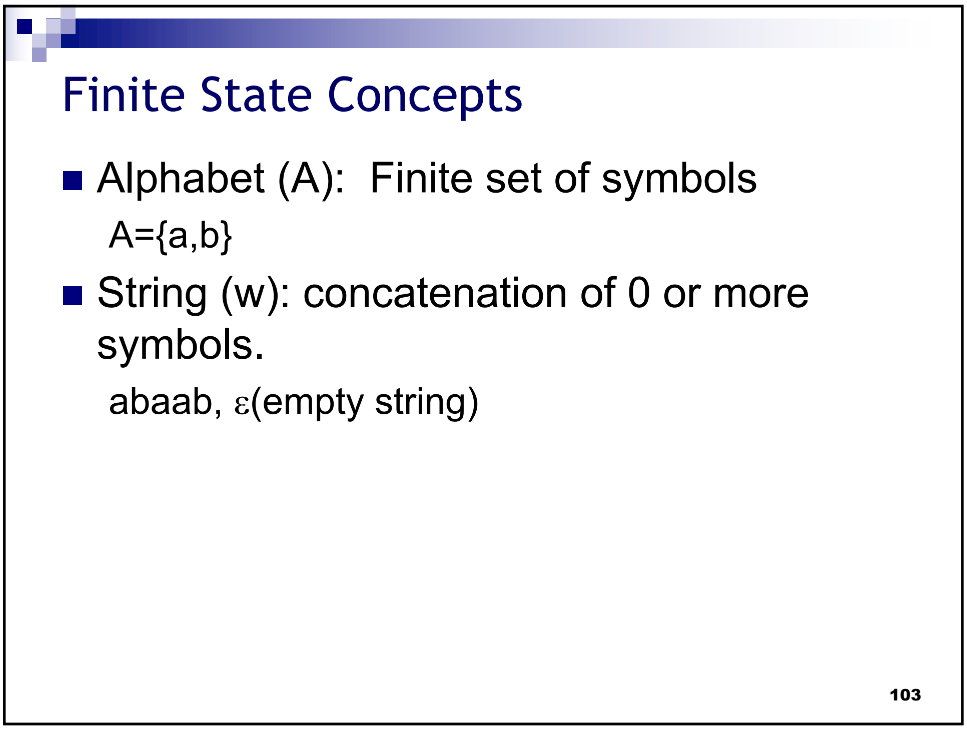
OVERVIEW

- Overview of Morphology
- Computational Morphology
- Overview of Finite State Machines
- Finite State Morphology
  - □ Two-level Morphology
  - □ Cascade Rules



Why is the Finite State Approach Interesting?

- Finite state systems are mathematically well-understood, elegant, flexible.
- Finite state systems are computationally efficient.
- For typical natural language processing tasks, finite state systems provide compact representations.
- Finite state systems are inherently bidirectional



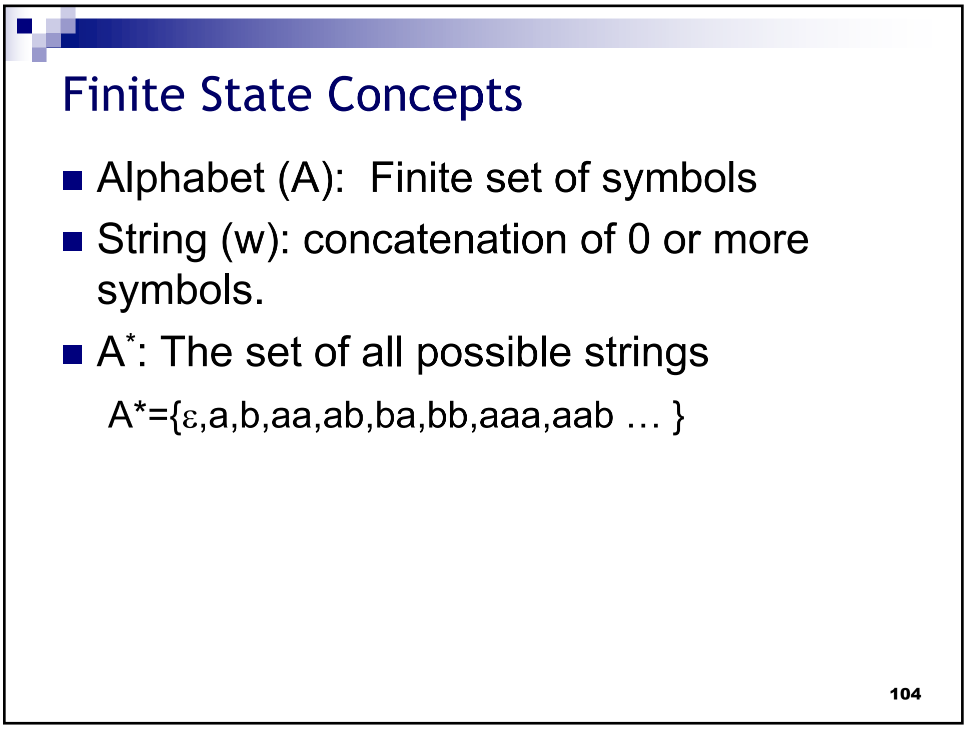
Finite State Concepts

- Alphabet (A): Finite set of symbols

  A={a,b}

- String (w): concatenation of 0 or more symbols.

  abaab, ε(empty string)



Finite State Concepts

- Alphabet (A): Finite set of symbols
- String (w): concatenation of 0 or more symbols.
- A^*: The set of all possible strings

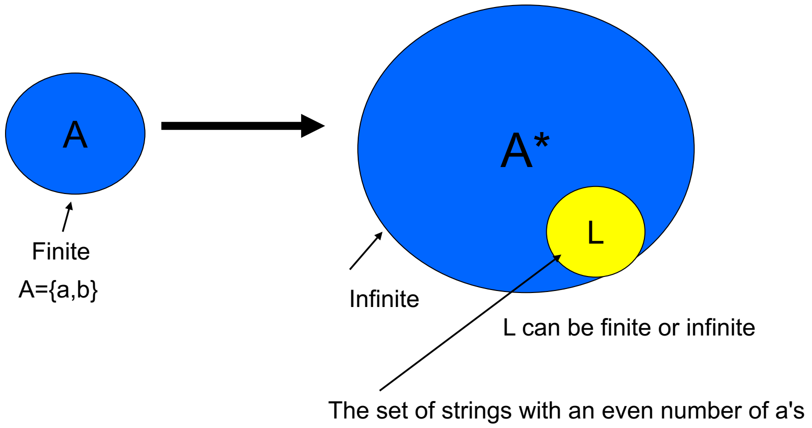
  A*={ε,a,b,aa,ab,ba,bb,aaa,aab … }

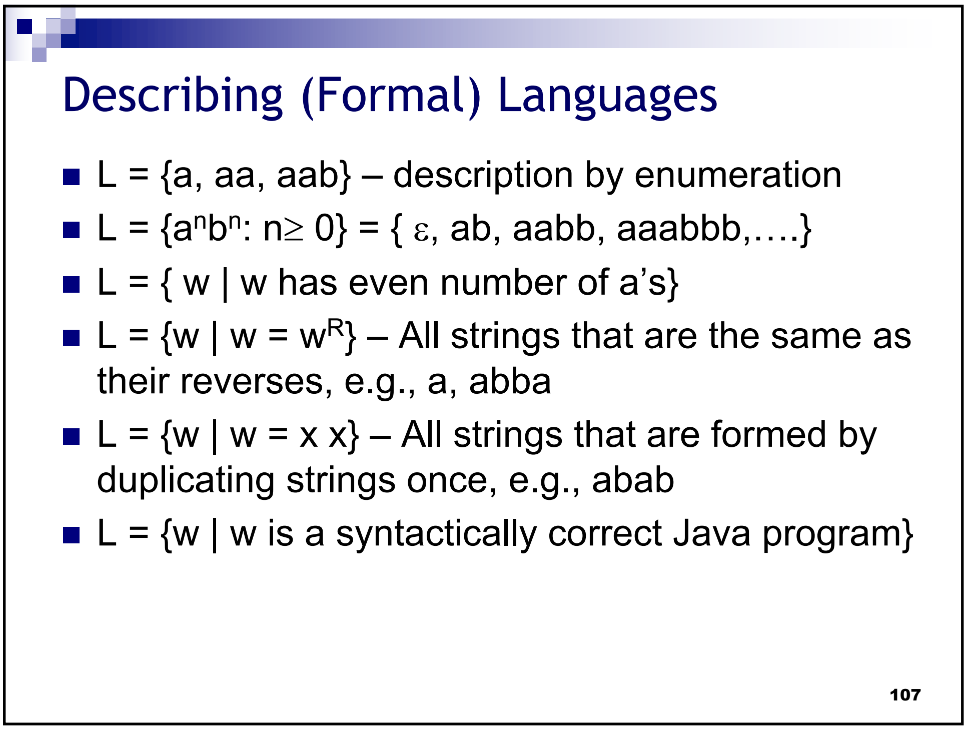
Finite State Concepts

- Alphabet (A): Finite set of symbols
- String (w): concatenation of 0 or more symbols.
- A^*: The set of all possible strings
- (formal) Language (L): a subset of A^*
  - □ The set of strings with an even number of a's
    - e.g. abbaaba but not abbaa
  - □ The set of strings with equal number of a's and b's
    - e.g., ababab but not bbbaa

Alphabets and Languages

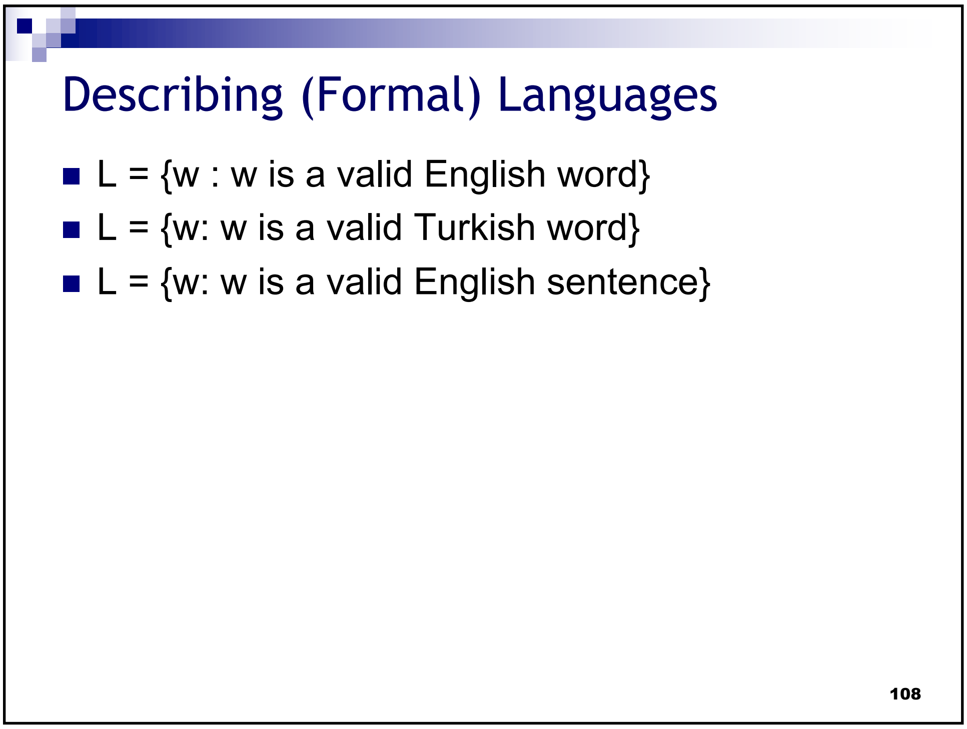
A*={ε,a,b,aa,ab,ba,bb,aaa,aab … }


A
A*
L
Finite
A={a,b}
Infinite
L can be finite or infinite
The set of strings with an even number of a's



Describing (Formal) Languages

- L = {a, aa, aab} – description by enumeration
- L = $\{a^n b^n : n \geq 0\} = \{ \varepsilon, ab, aabb, aaabbb, \ldots .\}$
- L = { w | w has even number of a’s}
- L = $\{w \mid w = w^R\}$ – All strings that are the same as their reverses, e.g., a, abba
- L = {w | w = x x} – All strings that are formed by duplicating strings once, e.g., abab
- L = {w | w is a syntactically correct Java program}



Describing (Formal) Languages

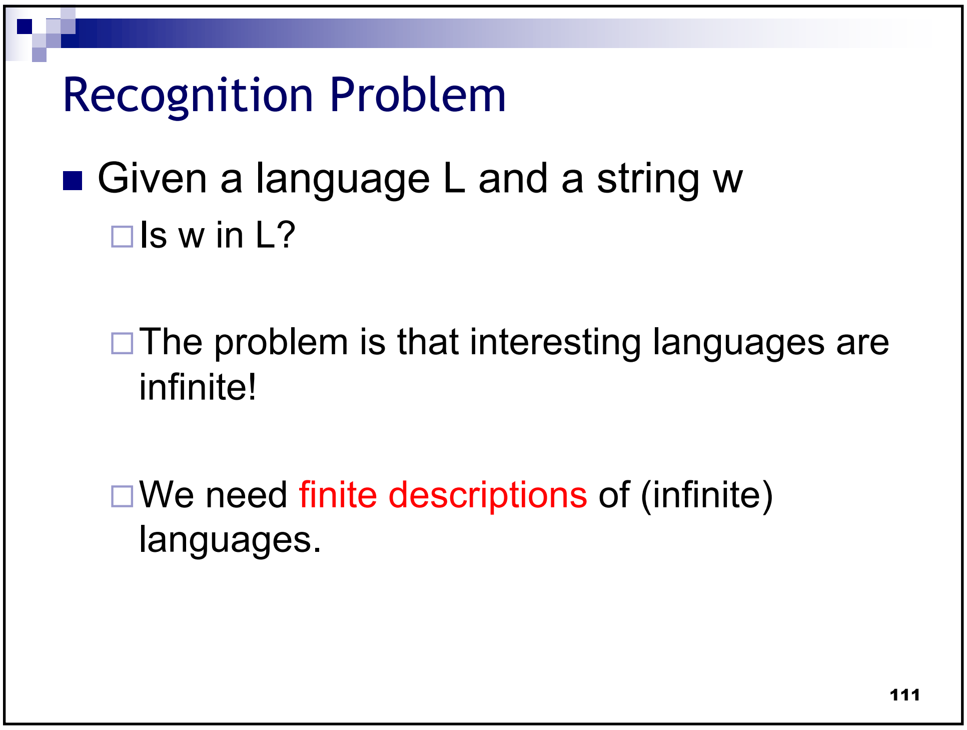
- L = {w : w is a valid English word}
- L = {w: w is a valid Turkish word}
- L = {w: w is a valid English sentence}

Languages

- Languages are sets. So we can do “set” things with them
  - □Union
  - □Intersection
  - □Complement with respect to the universe set A*.

Describing (Formal) Languages

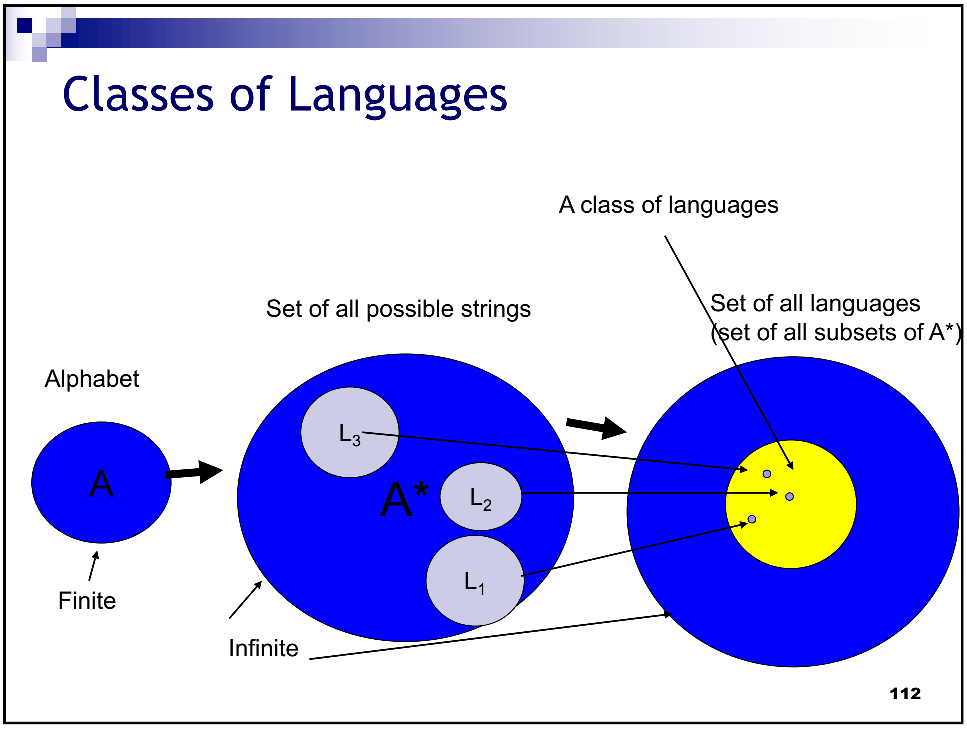
- How do we describe languages?
  - □L = $\{a^n b^n : n \geq 0\}$ is fine but not that terribly useful
- We need finite descriptions (of usually infinite sets)
- We need to be able to use these descriptions in “mechanizable” procedures.
  - □e.g., to check if a given string is in the language or not



Recognition Problem

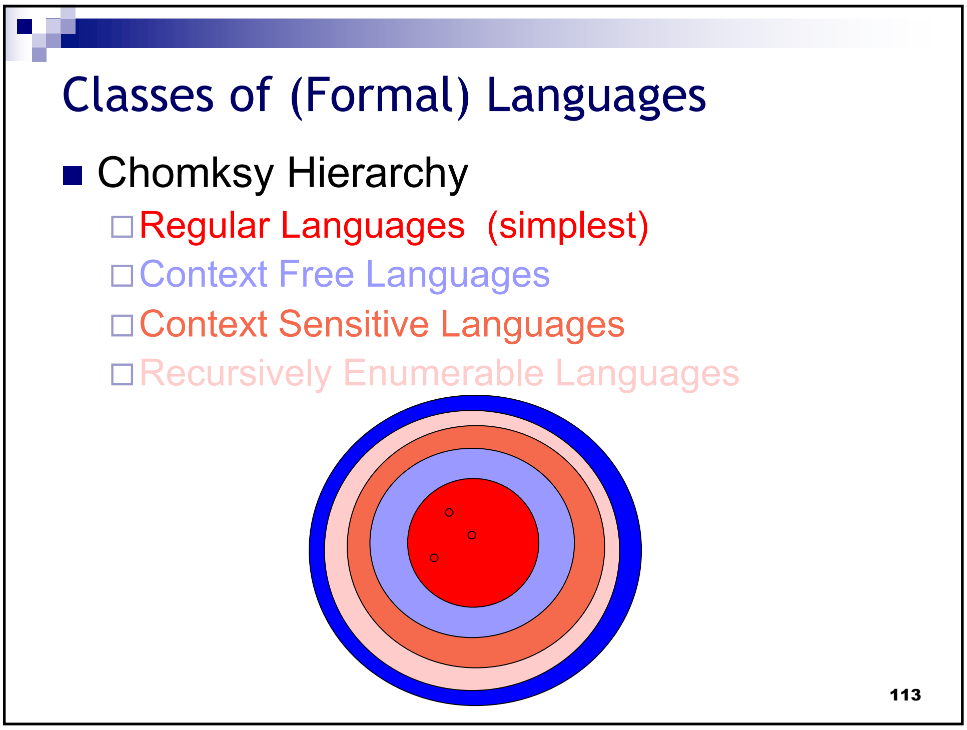
- Given a language L and a string w
  - □ Is w in L?
  - □ The problem is that interesting languages are infinite!
  - □ We need finite descriptions of (infinite) languages.

111



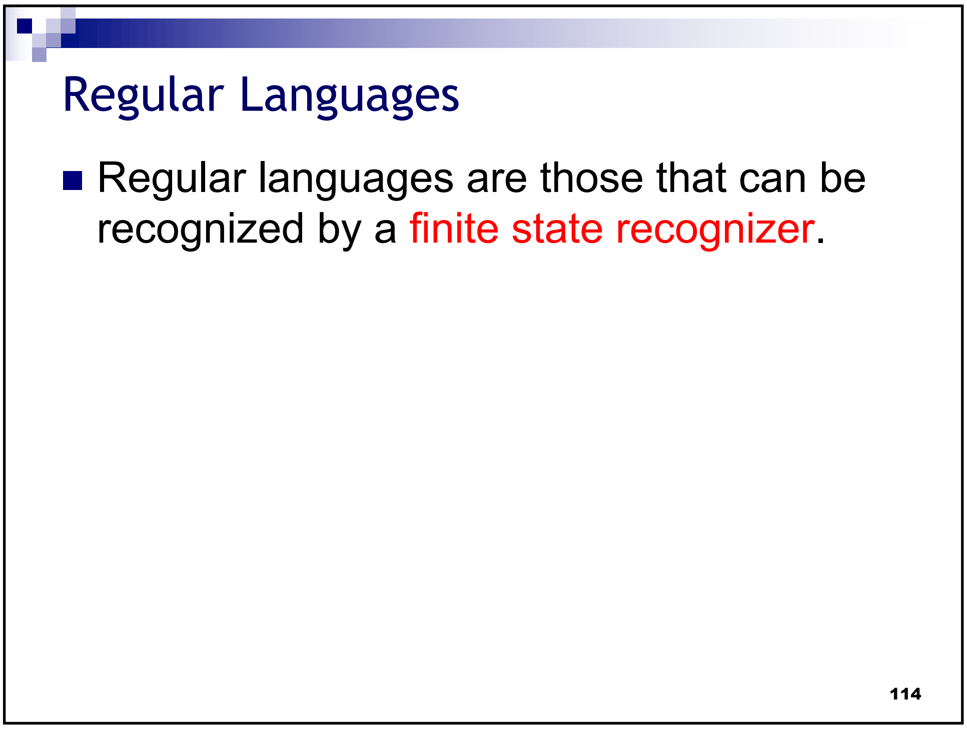
Classes of Languages

112



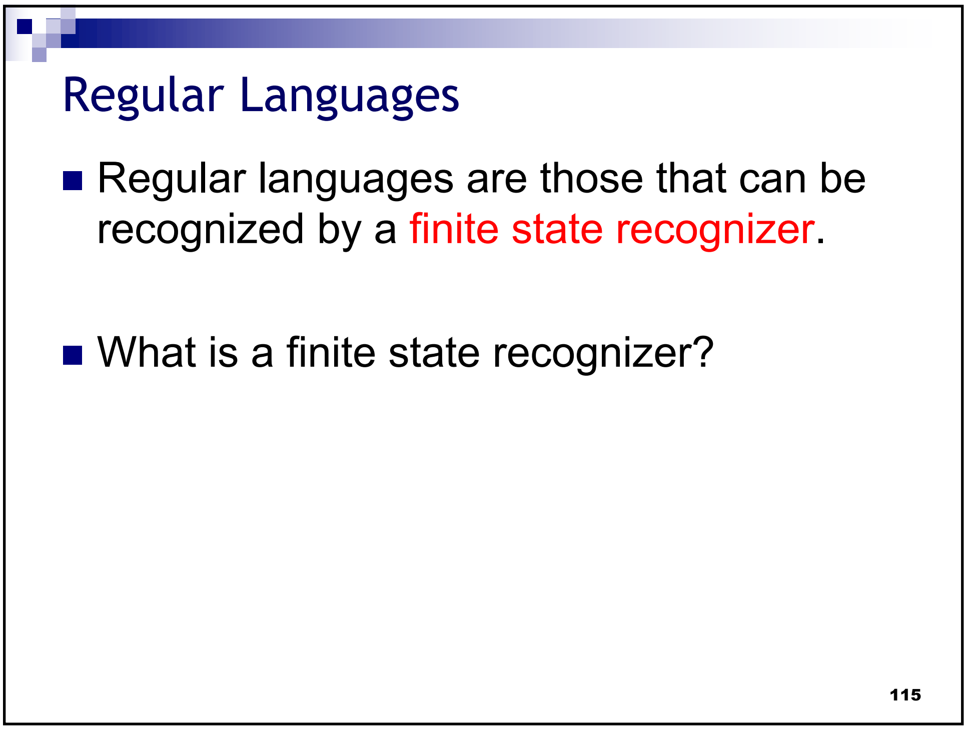
Classes of (Formal) Languages

- Chomksy Hierarchy
  - □ Regular Languages (simplest)
  - □ Context Free Languages
  - □ Context Sensitive Languages
  - □ Recursively Enumerable Languages



Regular Languages

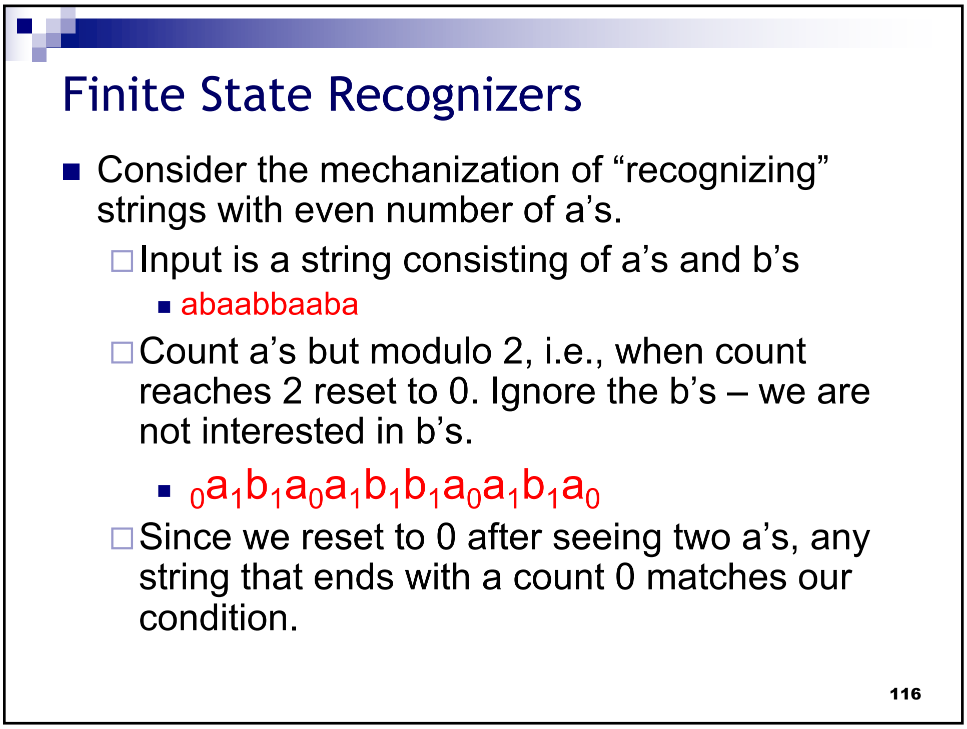
- Regular languages are those that can be recognized by a finite state recognizer.



Regular Languages

- Regular languages are those that can be recognized by a finite state recognizer.
- What is a finite state recognizer?

115



Finite State Recognizers

- Consider the mechanization of “recognizing” strings with even number of a’s.
  - □ Input is a string consisting of a’s and b’s
    - abaabbaaba
  - □ Count a’s but modulo 2, i.e., when count reaches 2 reset to 0. Ignore the b’s – we are not interested in b’s.
    - $_0a_1b_1a_0a_1b_1b_1a_0a_1b_1a_0$
  - □ Since we reset to 0 after seeing two a’s, any string that ends with a count 0 matches our condition.

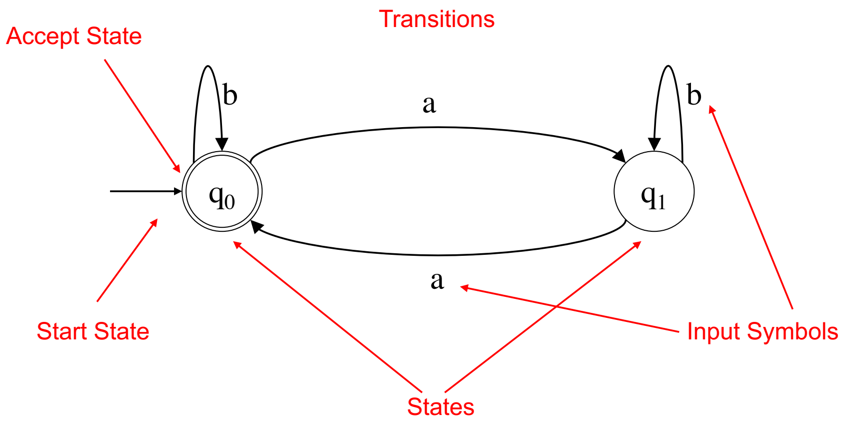
116

Finite State Recognizers

- $_0a_1b_1a_0a_1b_1b_1a_0a_1b_1a_0$
- The symbols shown in between input symbols encode/remember some “interesting” property of the string seen so far.
  - □ e.g., a 0 shows that we have seen an even number of a’s, while a 1, shows an odd number of a’s, so far.
- Let’s call this “what we remember from past” as the state.
- A “finite state recognizer” can only remember a finite number of distinct pieces of information from the past.

Finite State Recognizers

- We picture FSRs with state graphs.



Transitions

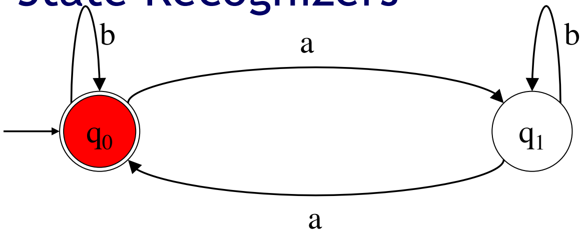
Accept State

Start State

Input Symbols

States

Finite State Recognizers

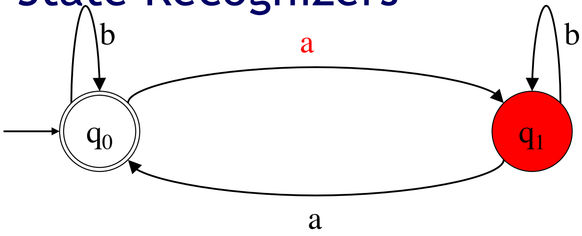


Is abab in the language?

a b a b

^

Finite State Recognizers

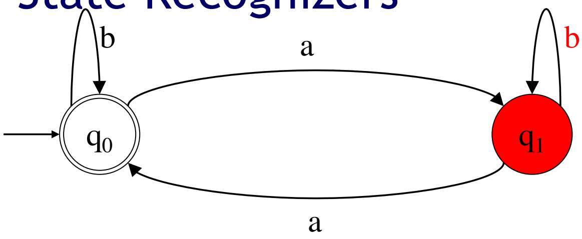


Is abab in the language?

a b a b

^

Finite State Recognizers

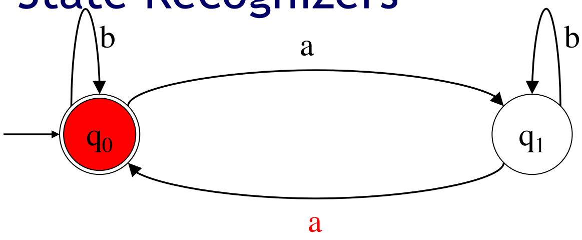


Is abab in the language?

a b a b
  ∧

121

Finite State Recognizers

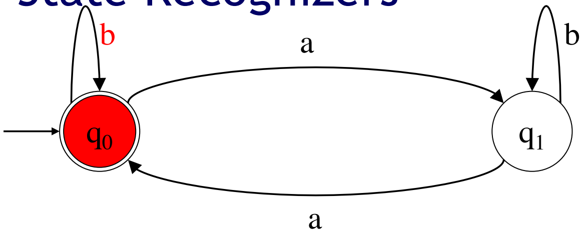


Is abab in the language?

a b a b
    ∧

122

Finite State Recognizers

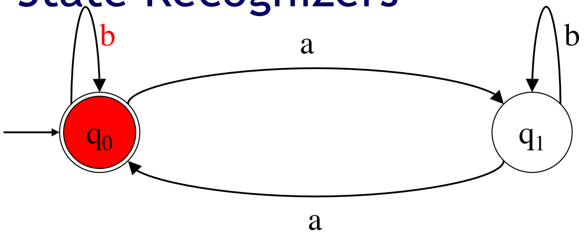


Is abab in the language? YES

a b a b

^

Finite State Recognizers

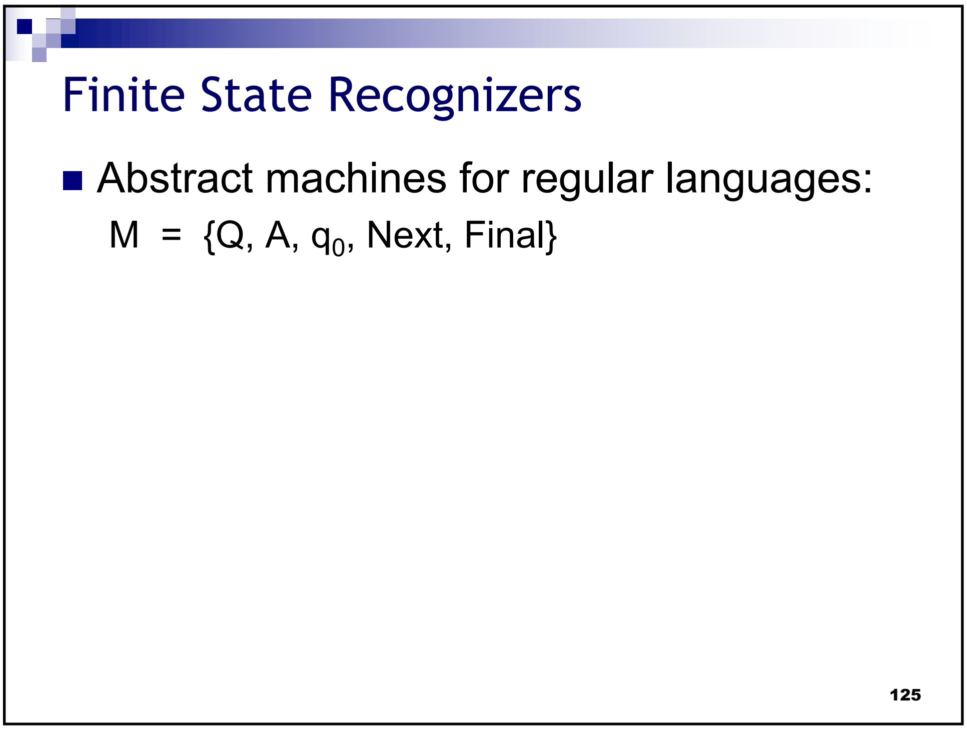


Is abab in the language? YES

a b a b

^

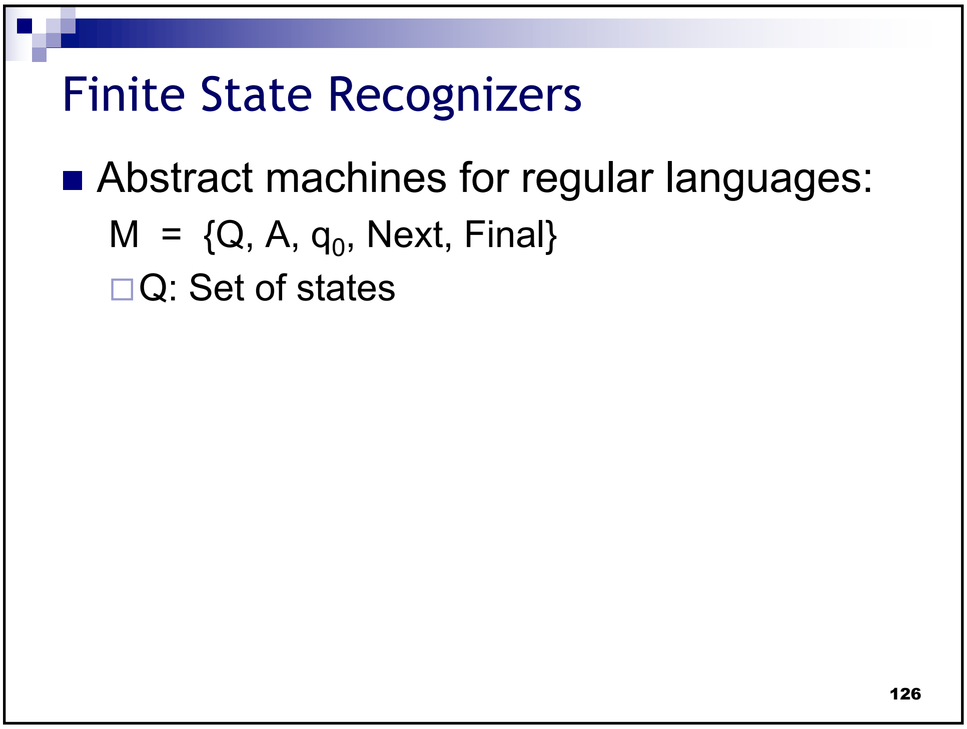
The state q_0 remembers the fact that we have seen an even number of a's
The state q_1 remembers the fact that we have seen an odd number of a's



Finite State Recognizers

- Abstract machines for regular languages:

  M = {Q, A, q_0, Next, Final}

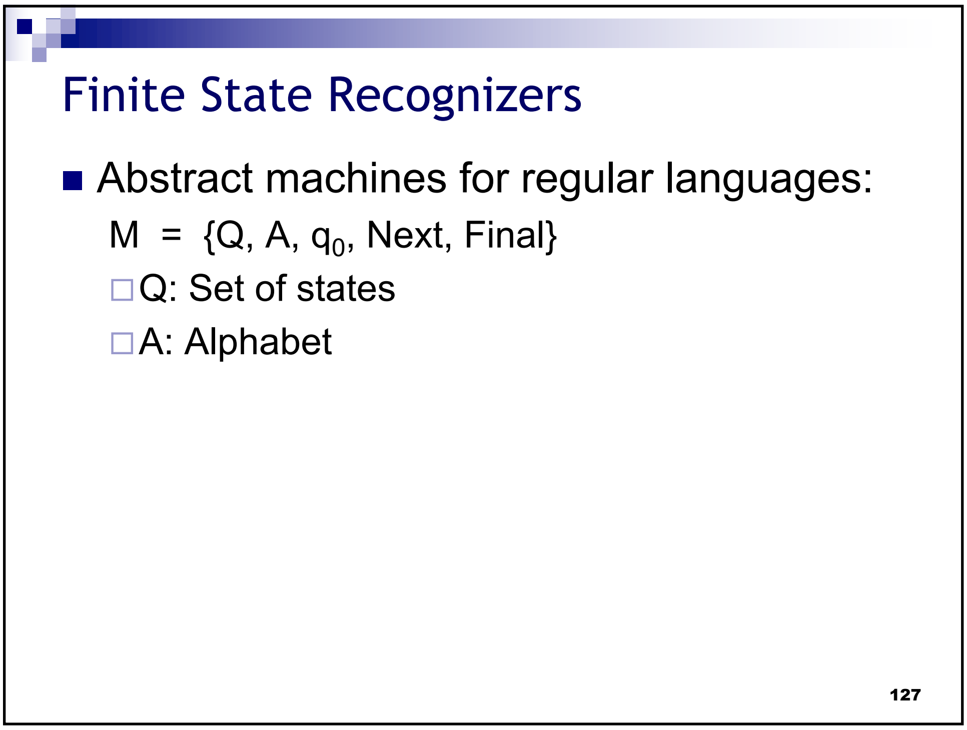


Finite State Recognizers

- Abstract machines for regular languages:

  M = {Q, A, q_0, Next, Final}

  - □Q: Set of states



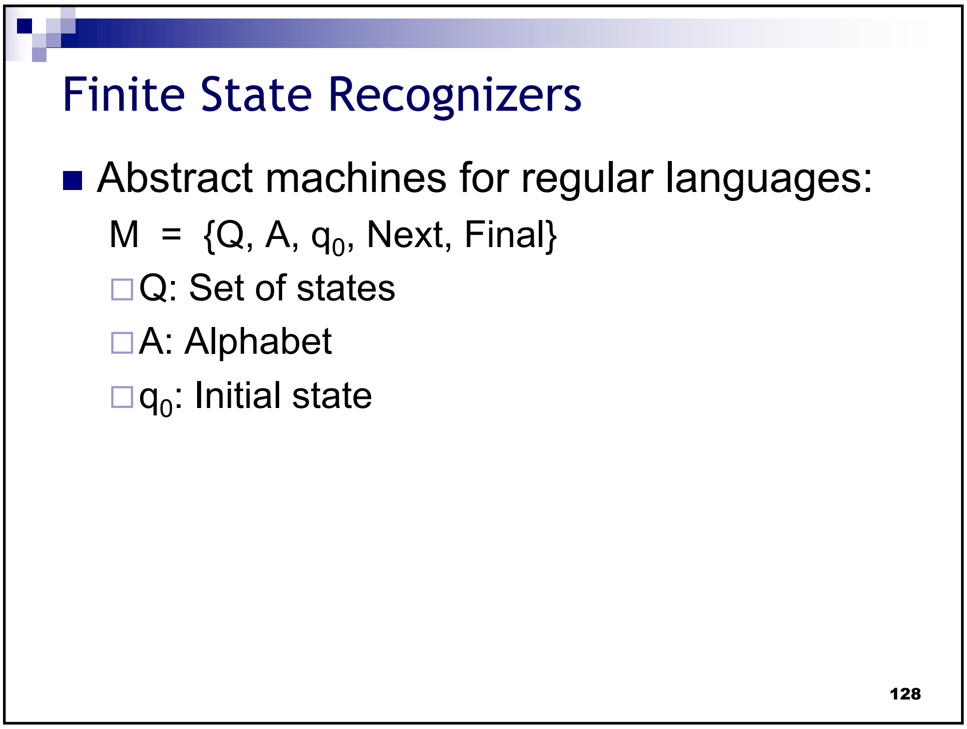
Finite State Recognizers

- Abstract machines for regular languages:

  M = {Q, A, q_0, Next, Final}

  □ Q: Set of states

  □ A: Alphabet



Finite State Recognizers

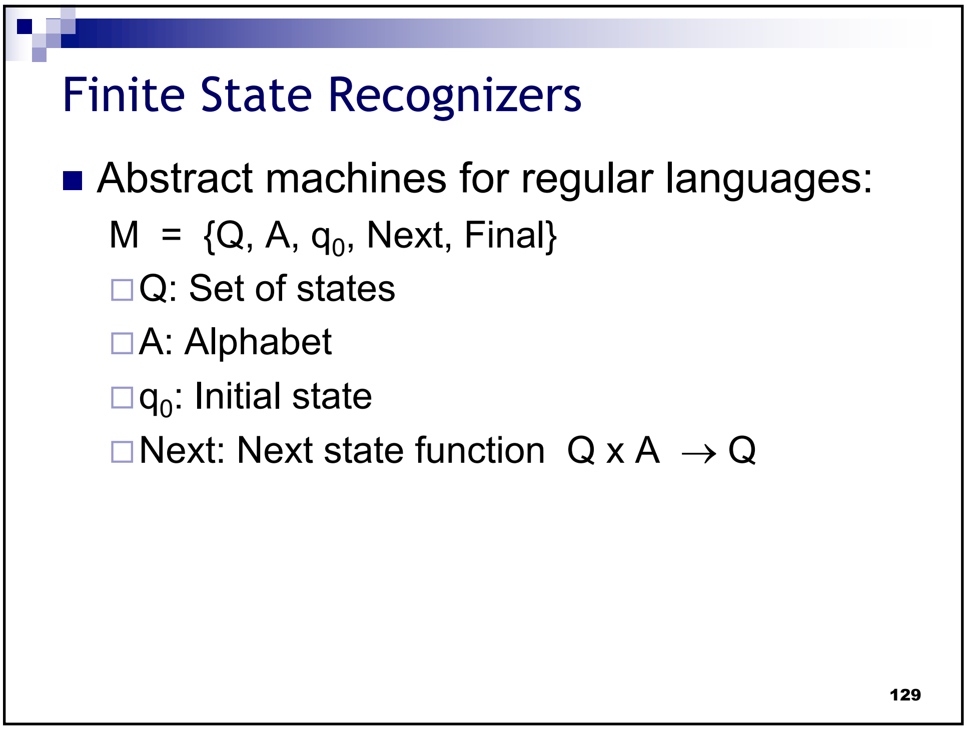
- Abstract machines for regular languages:

  M = {Q, A, q_0, Next, Final}

  □ Q: Set of states

  □ A: Alphabet

  □ q_0: Initial state

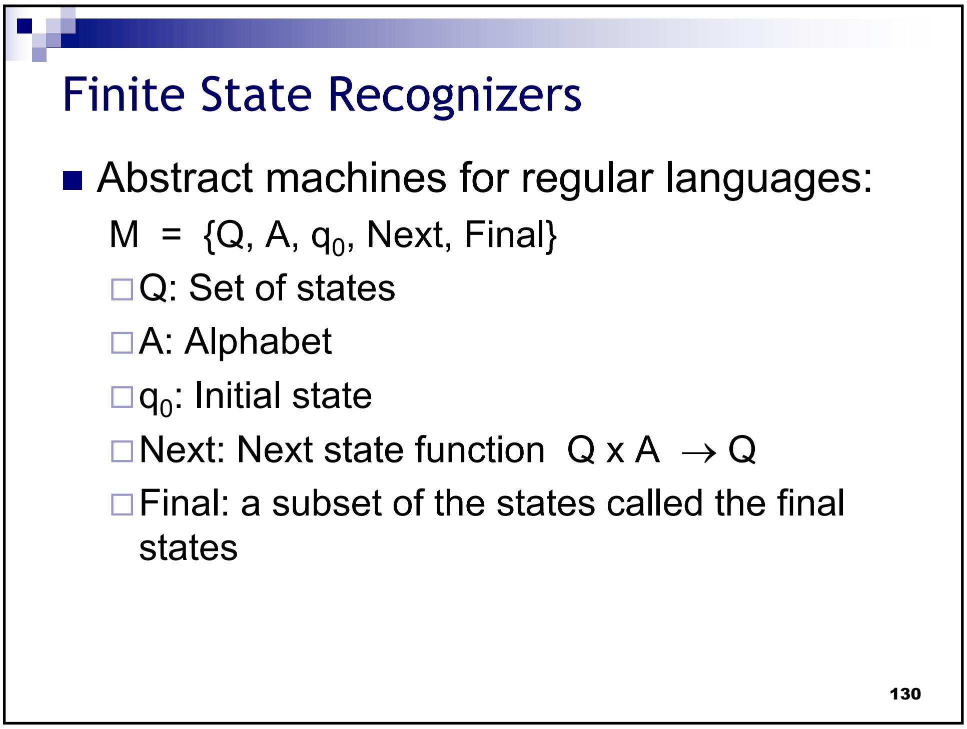


Finite State Recognizers

- Abstract machines for regular languages:

  M = {Q, A, q_0, Next, Final}

  - □ Q: Set of states
  - □ A: Alphabet
  - □ q_0: Initial state
  - □ Next: Next state function Q x A → Q



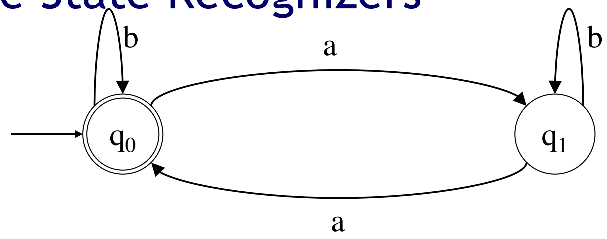
Finite State Recognizers

- Abstract machines for regular languages:

  M = {Q, A, q_0, Next, Final}

  - □ Q: Set of states
  - □ A: Alphabet
  - □ q_0: Initial state
  - □ Next: Next state function Q x A → Q
  - □ Final: a subset of the states called the final states

Finite State Recognizers



$A = \{a,b\}$

$Q = \{q_0, q_1\}$

$\text{Next} = \{((q_0,b),q_0),$
$((q_0,a),q_1), \rightarrow$
$((q_1,b),q_1),$
$((q_1,a),q_0))\}$

$\text{Final} = \{q_0\}$

If the machine is in state **q_0** and the input is **a** then the next state is **q_1**

Finite State Recognizers

- Abstract machines for regular languages:

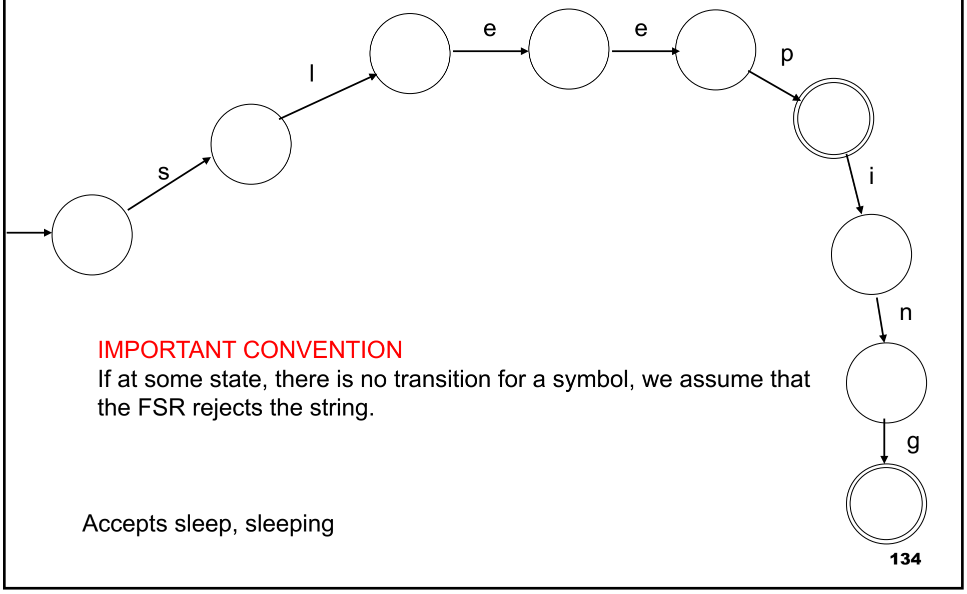
  M = {Q, A, q_0, Next, Final}

- M accepts w $\in$ A*, if
  - □ starting in state q_0, M proceeds by looking at each symbol in w, and
  - □ ends up in one of the final states when the string w is exhausted.

Finite State Recognizers

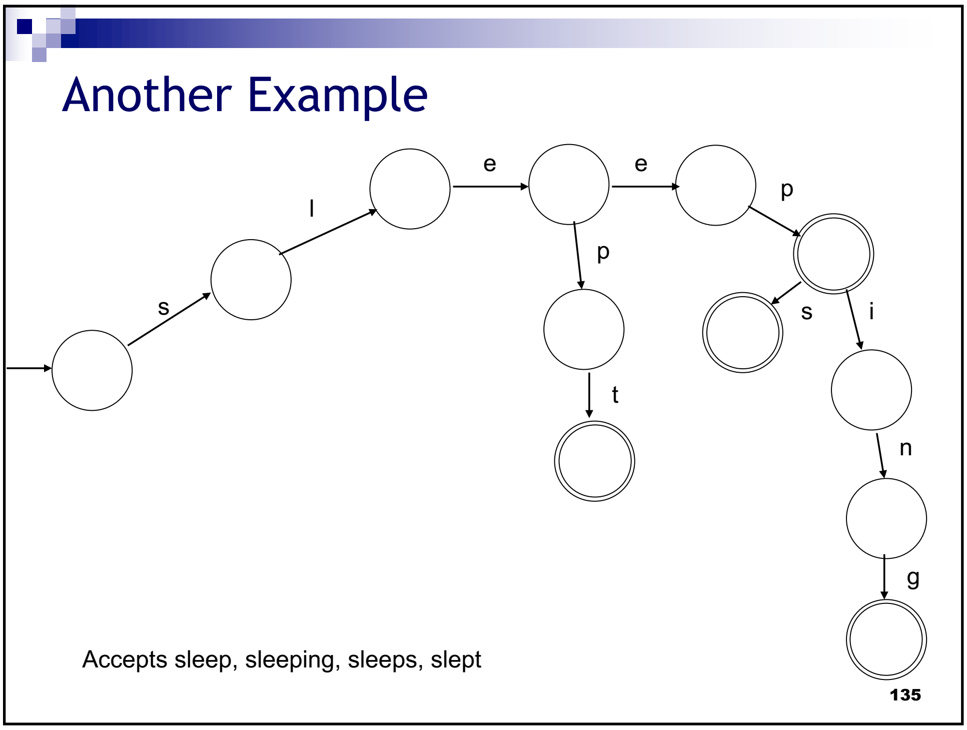
- Note that the description of the recognizer itself is finite.
  - □ Finite number of states
  - □ Finite number of alphabet symbols
  - □ Finite number of transitions
- Number of strings accepted can be infinite.

Another Example



IMPORTANT CONVENTION
If at some state, there is no transition for a symbol, we assume that the FSR rejects the string.

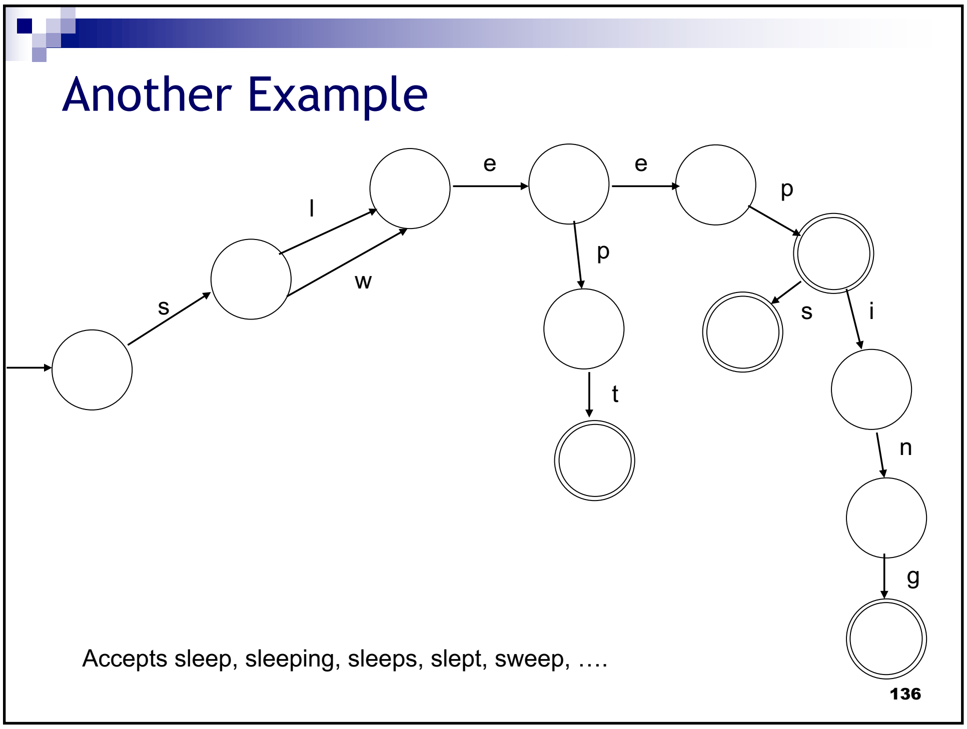
Accepts sleep, sleeping



Another Example

Accepts sleep, sleeping, sleeps, slept

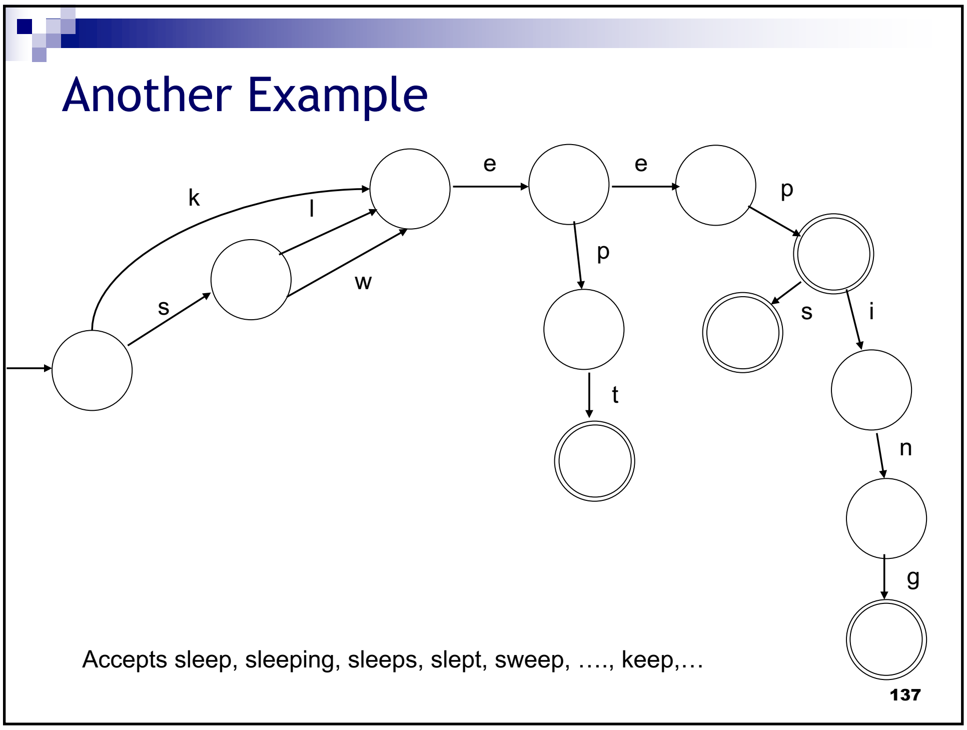
135



Another Example

Accepts sleep, sleeping, sleeps, slept, sweep, ….

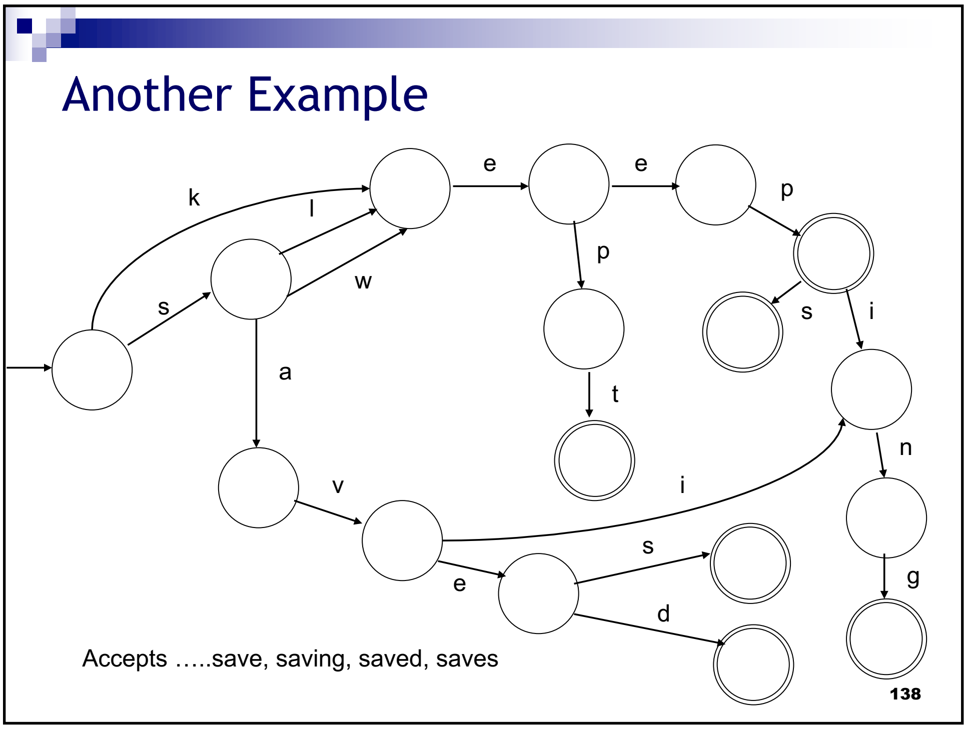
136



Another Example

Accepts sleep, sleeping, sleeps, slept, sweep, …., keep,…

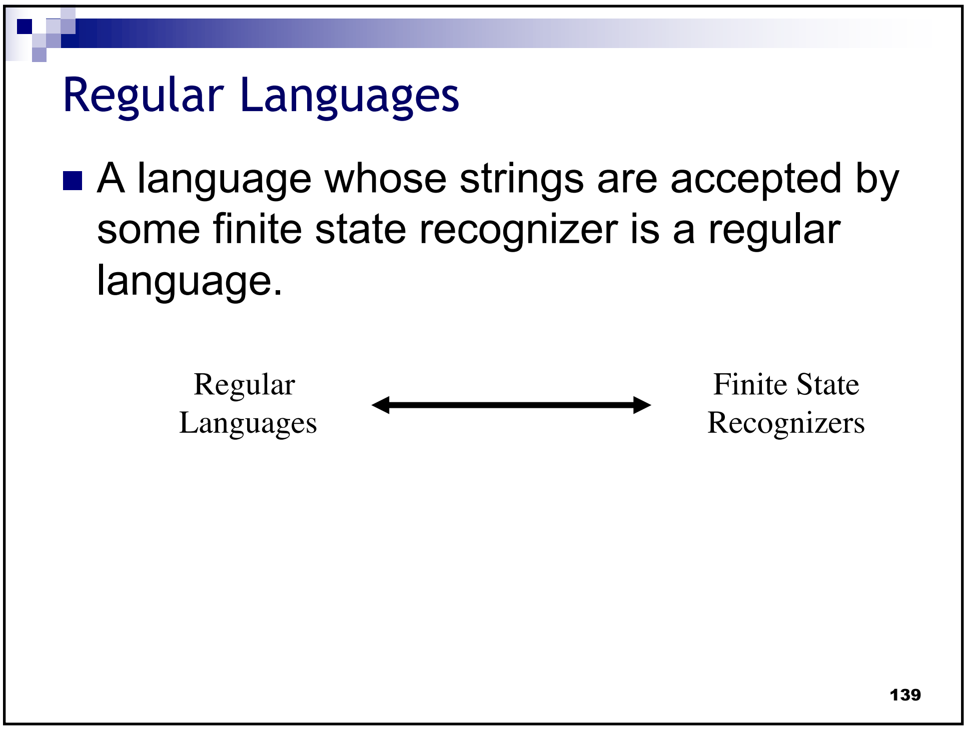
137



Another Example

Accepts …..save, saving, saved, saves

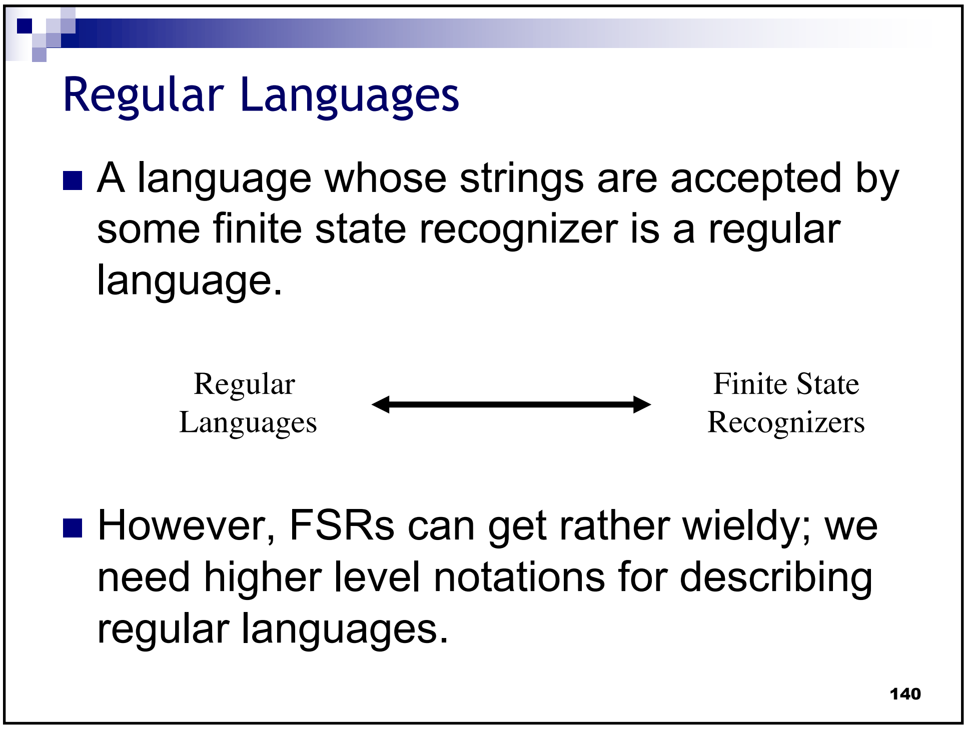
138



Regular Languages

- A language whose strings are accepted by some finite state recognizer is a regular language.

Regular Languages ⟷ Finite State Recognizers

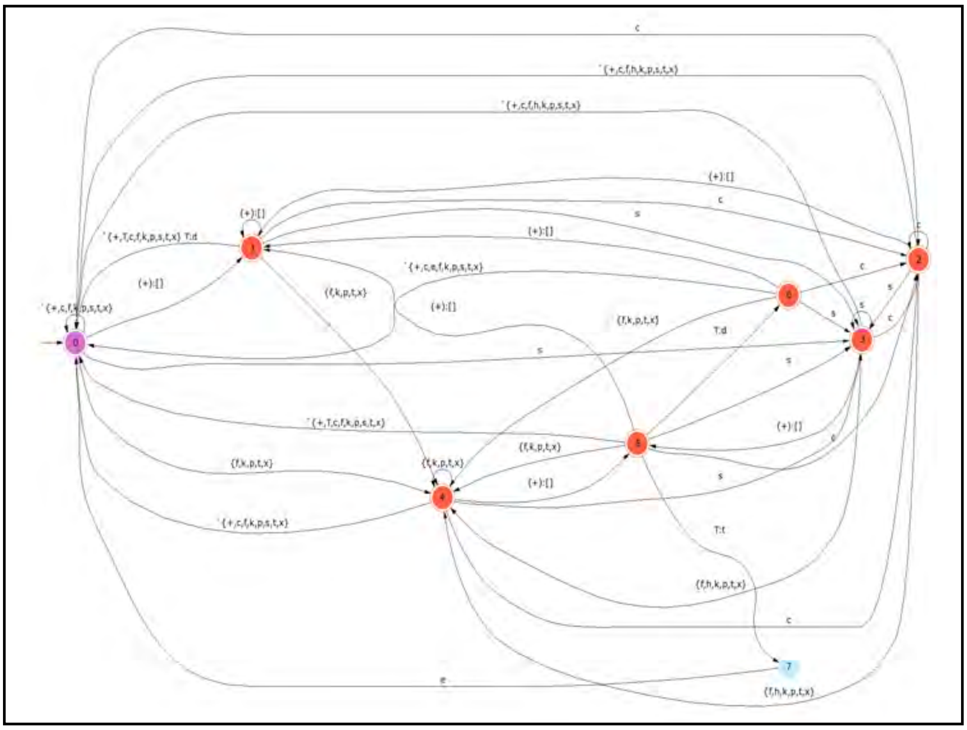


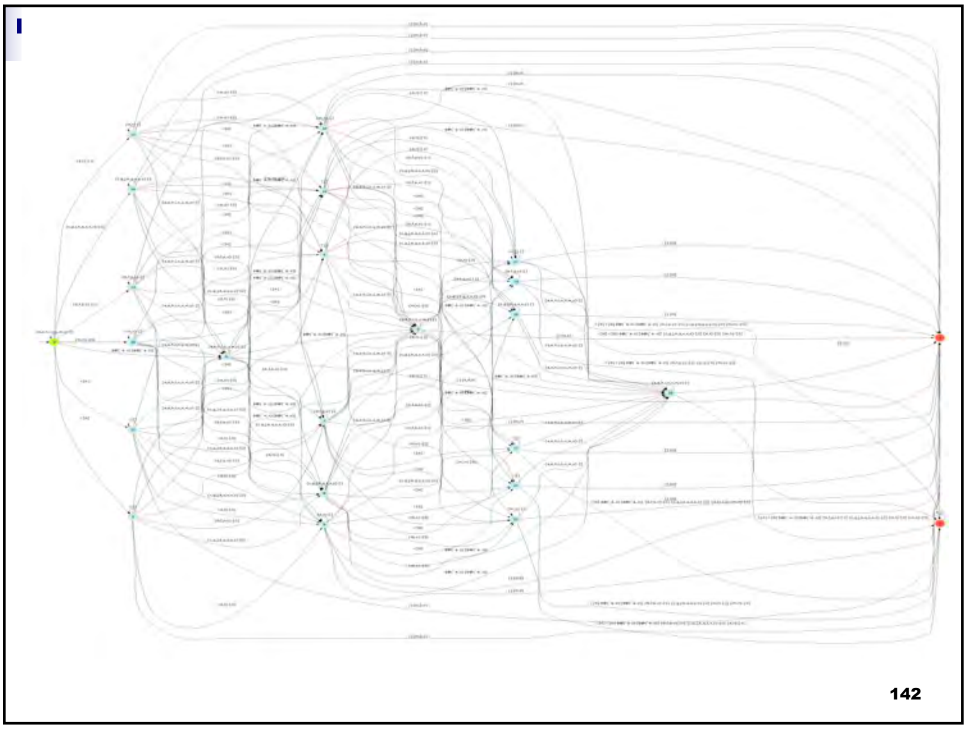
Regular Languages

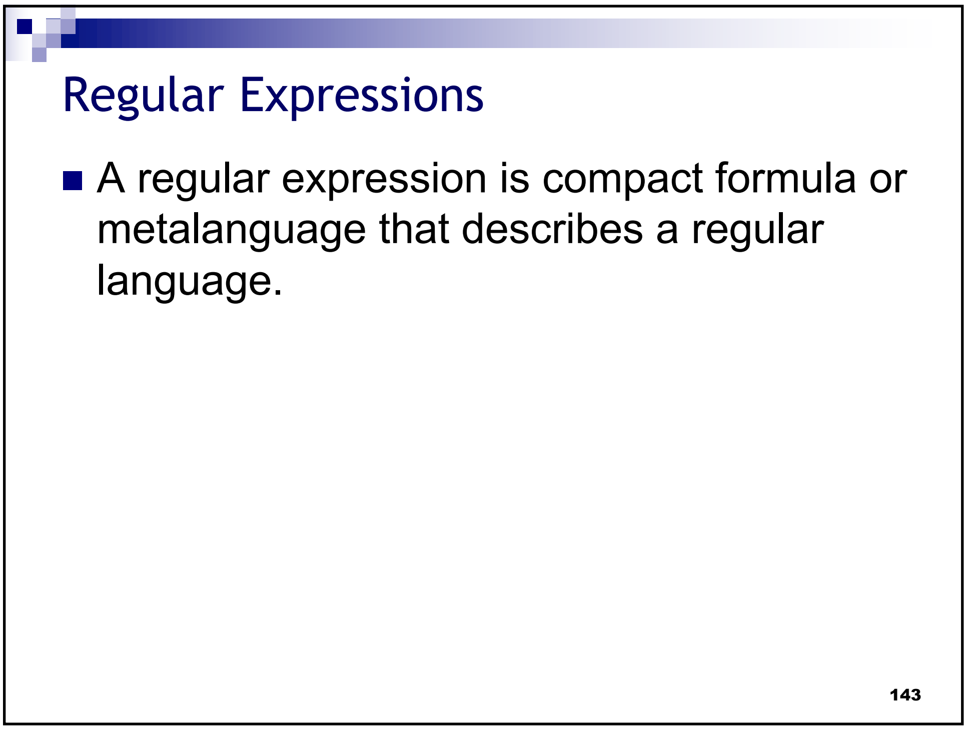
- A language whose strings are accepted by some finite state recognizer is a regular language.

Regular Languages ⟷ Finite State Recognizers

- However, FSRs can get rather wieldy; we need higher level notations for describing regular languages.


c
^{+,c,f,h,k,p,s,t,x}
^{+,c,f,h,k,p,s,t,x}
(+):[]
c
s
(+):[]
^{+,T,c,f,k,p,s,t,x} T:d
(+):[]
^{+,c,f,k,p,s,t,x}
{f,k,p,t,x}
^{+,c,e,f,k,p,s,t,x}
(+):[]
{f,k,p,t,x}
c
s
T:d
s
s
s
c
^{+,T,c,f,k,p,e,t,x}
(+):[]
{f,k,p,t,x}
{f,k,p,t,x}
{f,k,p,t,x}
(+):[]
s
^{+,c,f,k,p,s,t,x}
T:t
{f,h,k,p,t,x}
c
e
{f,h,k,p,t,x}
0
1
2
3
4
5
6
7

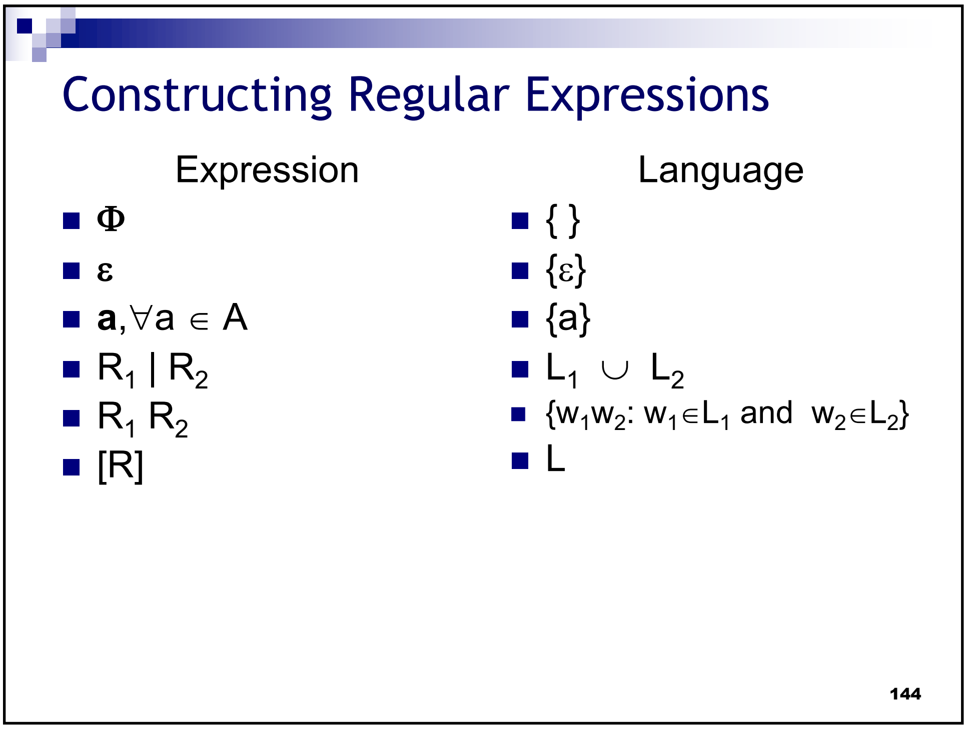




Regular Expressions

- A regular expression is compact formula or metalanguage that describes a regular language.

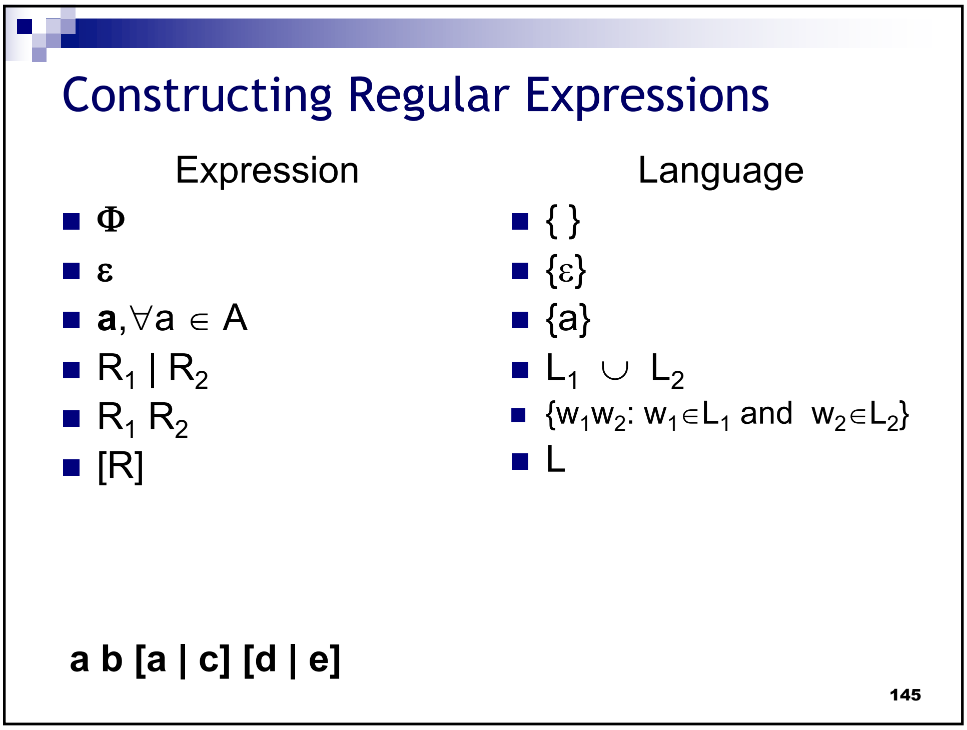
143



Constructing Regular Expressions

Expression	Language
Φ	{ }
ε	$\{\varepsilon\}$
a, $\forall a \in A$	{a}
$R_1 \mid R_2$	$L_1 \cup L_2$
$R_1 R_2$	$\{w_1w_2: w_1 \in L_1$ and $w_2 \in L_2\}$
[R]	L

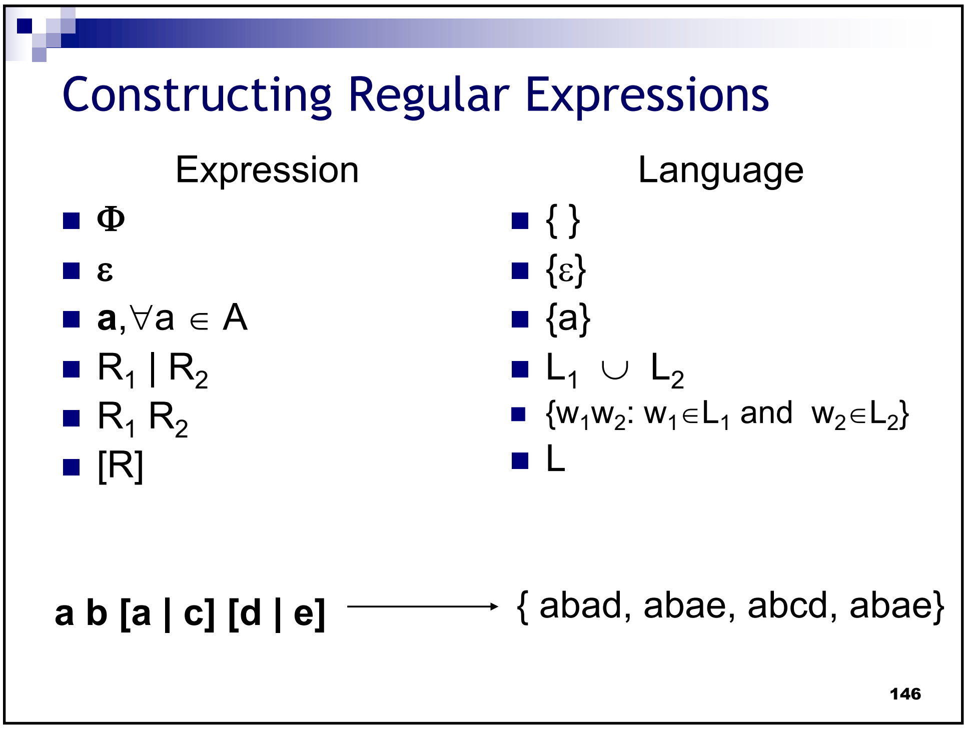
144



Constructing Regular Expressions

Expression	Language
Φ	{ }
ε	$\{\varepsilon\}$
$\mathbf{a}, \forall a \in A$	{a}
$R_1 \mid R_2$	$L_1 \cup L_2$
$R_1 R_2$	$\{w_1w_2: w_1 \in L_1 \text{ and } w_2 \in L_2\}$
[R]	L

a b [a | c] [d | e]



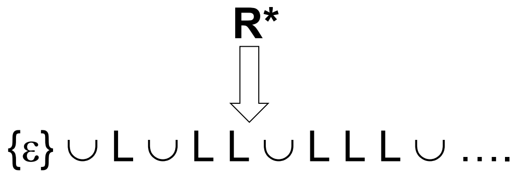
Constructing Regular Expressions

Expression	Language
Φ	{ }
ε	$\{\varepsilon\}$
$\mathbf{a}, \forall a \in A$	{a}
$R_1 \mid R_2$	$L_1 \cup L_2$
$R_1 R_2$	$\{w_1w_2: w_1 \in L_1 \text{ and } w_2 \in L_2\}$
[R]	L

a b [a | c] [d | e] → { abad, abae, abcd, abae}

Constructing Regular Expressions

- This much is not that interesting; allows us to describe only finite languages.
- The Kleene Star operator describes infinite sets.

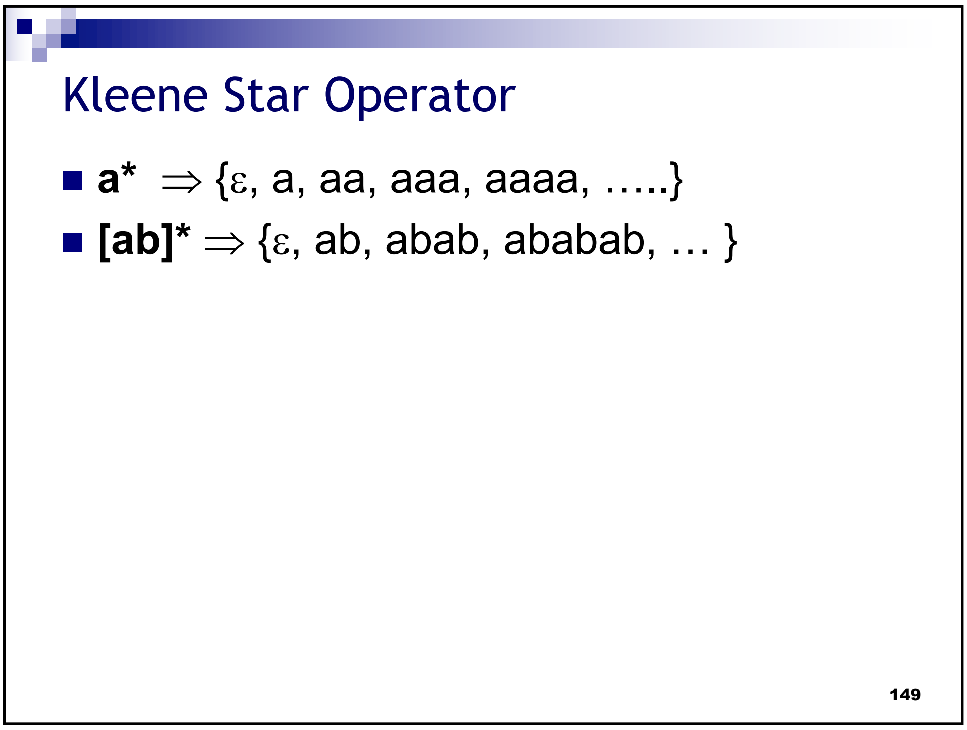


R*

$\{\varepsilon\} \cup L \cup L\,L \cup L\,L\,L \cup \ldots.$

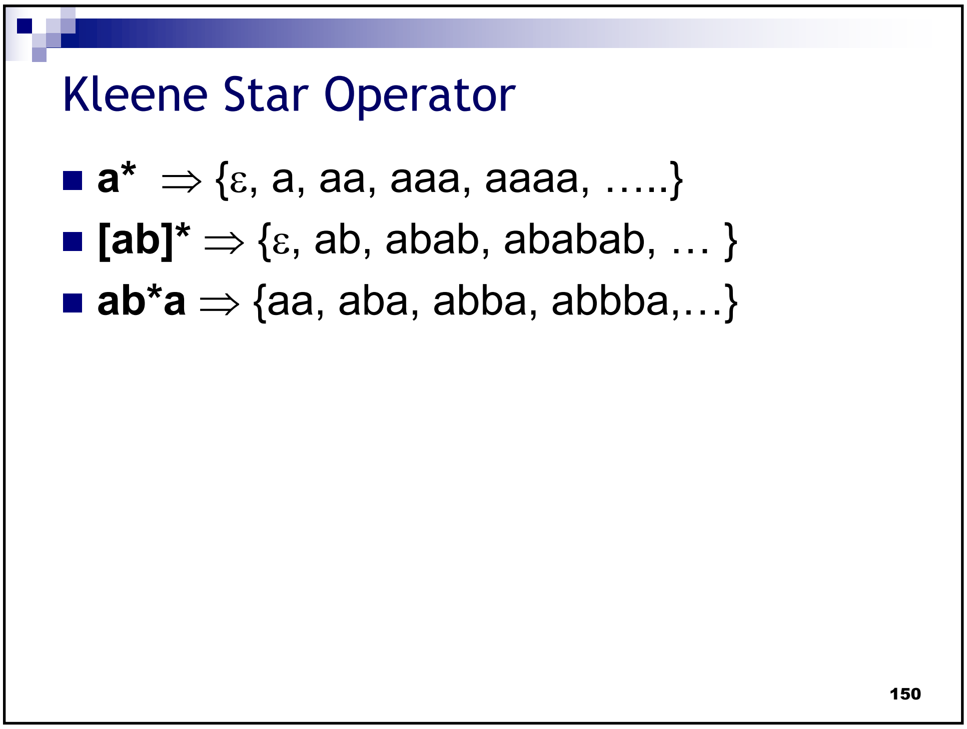
Kleene Star Operator

- **a*** $\Rightarrow \{\varepsilon, a, aa, aaa, aaaa, \ldots..\}$



Kleene Star Operator

- **a*** ⇒ {ε, a, aa, aaa, aaaa, …..}
- **[ab]*** ⇒ {ε, ab, abab, ababab, … }



Kleene Star Operator

- **a*** ⇒ {ε, a, aa, aaa, aaaa, …..}
- **[ab]*** ⇒ {ε, ab, abab, ababab, … }
- **ab*a** ⇒ {aa, aba, abba, abbba,…}

Kleene Star Operator

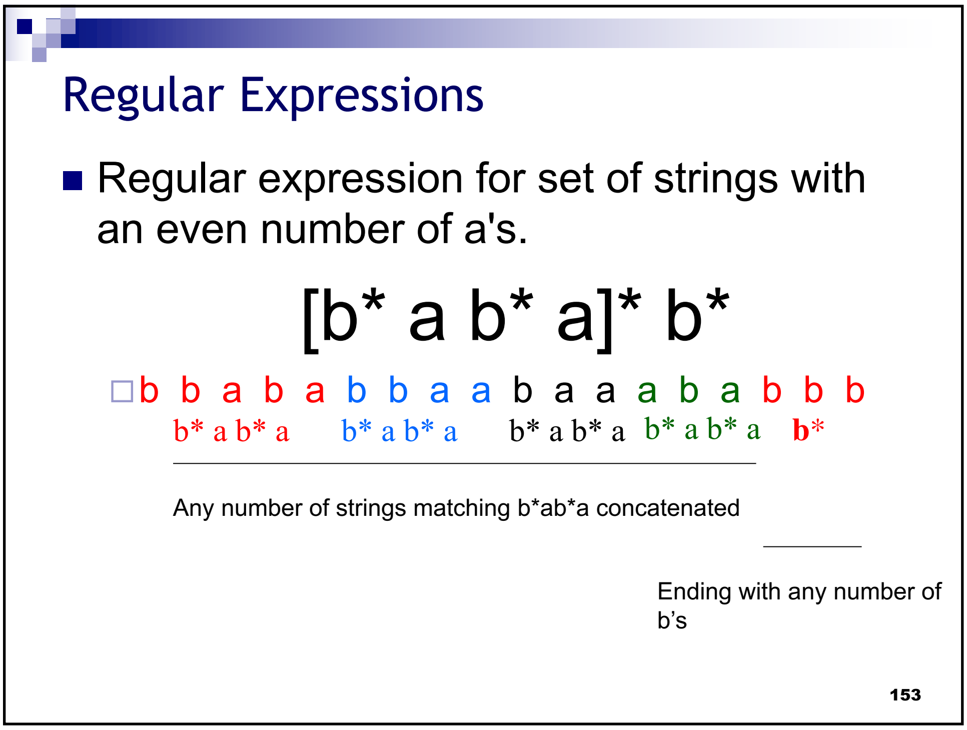
- **a*** ⇒ {ε, a, aa, aaa, aaaa, …..}
- **[ab]*** ⇒ {ε, ab, abab, ababab, … }
- **ab*a** ⇒ {aa, aba, abba, abbba,…}
- **[a|b]*[c|d]*** ⇒ {ε, a, b, c, d, ac, ad, bc, bd, aac, abc,bac, bbc, …..}

Regular Expressions

- Regular expression for set of strings with an even number of a's.

[b* a b* a]* b*

- □ Any number of concatenations of strings of the sort
  - Any number of b's followed by an a followed by any number of b's followed by another a
- □ Ending with any number of b's



Regular Expressions

- Regular expression for set of strings with an even number of a's.

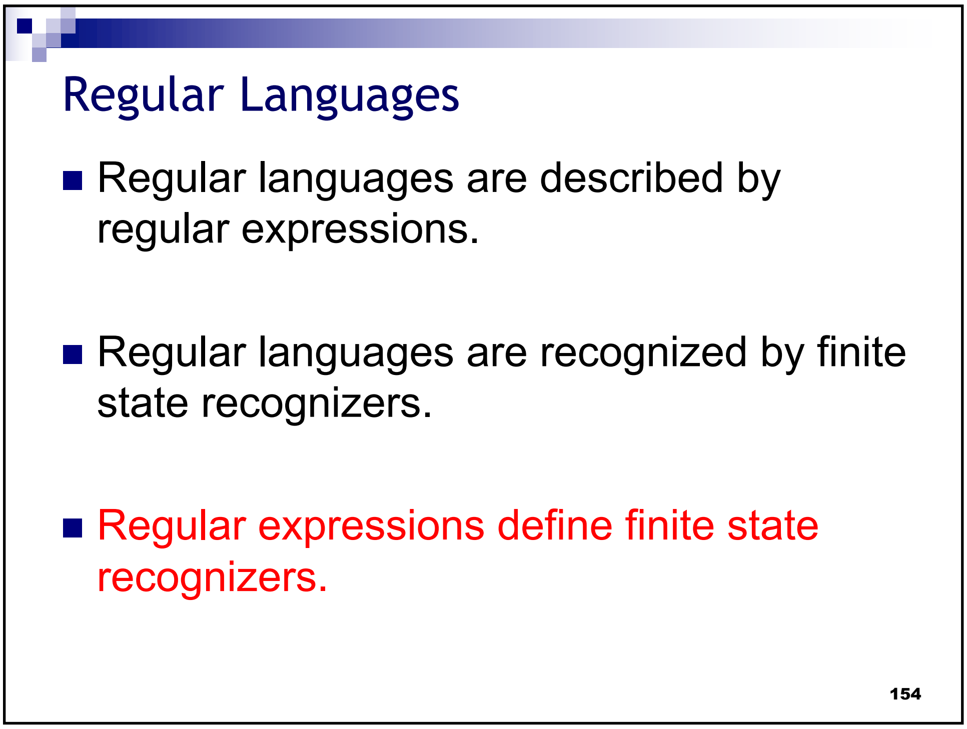
[b* a b* a]* b*

□b b a b a b b a a b a a a b a b b b

b* a b* a   b* a b* a   b* a b* a b* a b* a   b*

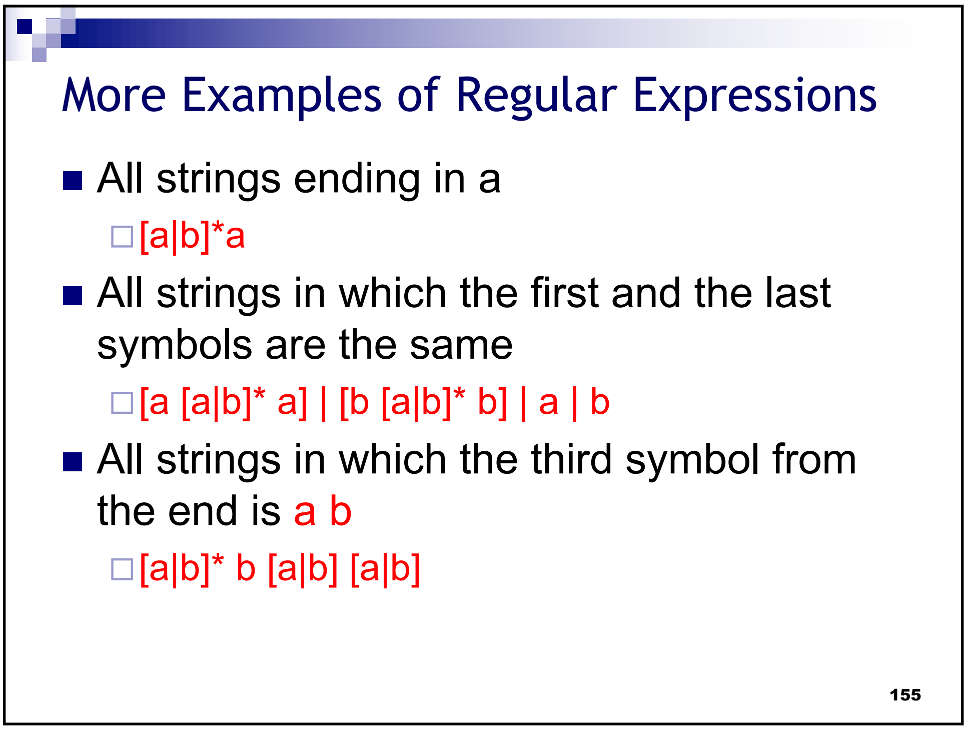
Any number of strings matching b*ab*a concatenated

Ending with any number of b's



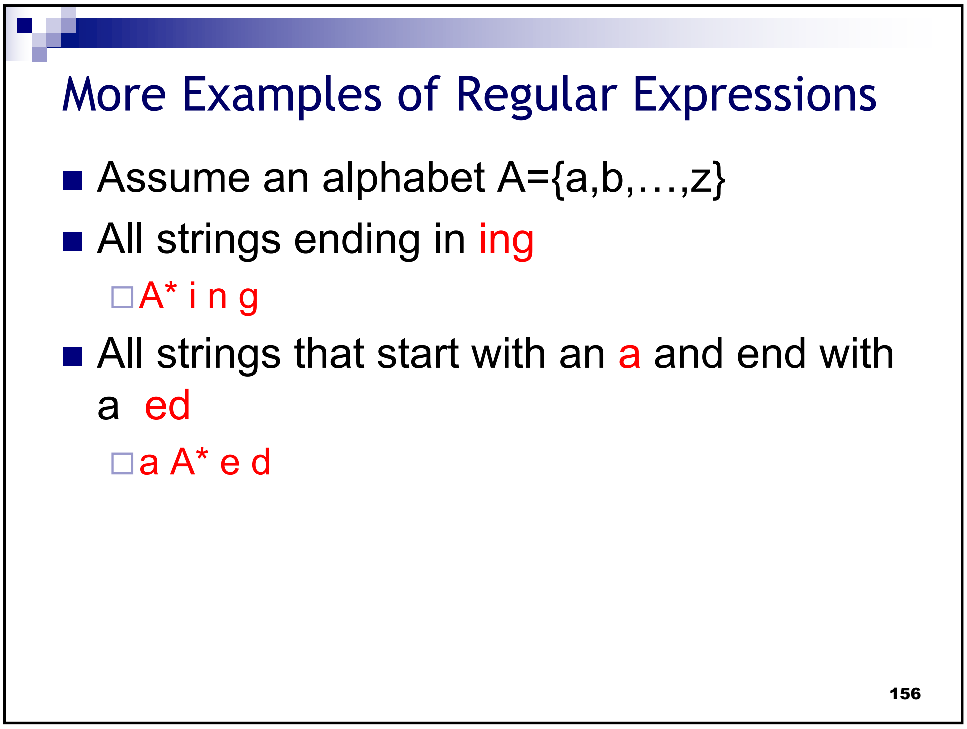
Regular Languages

- Regular languages are described by regular expressions.
- Regular languages are recognized by finite state recognizers.
- Regular expressions define finite state recognizers.



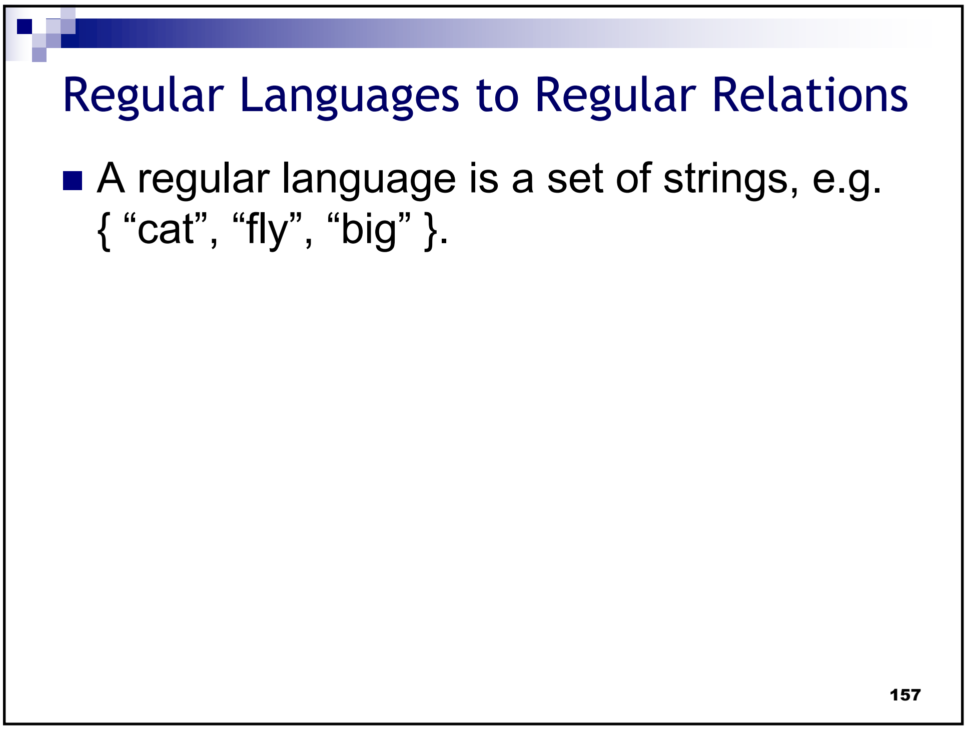
More Examples of Regular Expressions

- All strings ending in a
  - [a|b]*a
- All strings in which the first and the last symbols are the same
  - [a [a|b]* a] | [b [a|b]* b] | a | b
- All strings in which the third symbol from the end is a b
  - [a|b]* b [a|b] [a|b]



More Examples of Regular Expressions

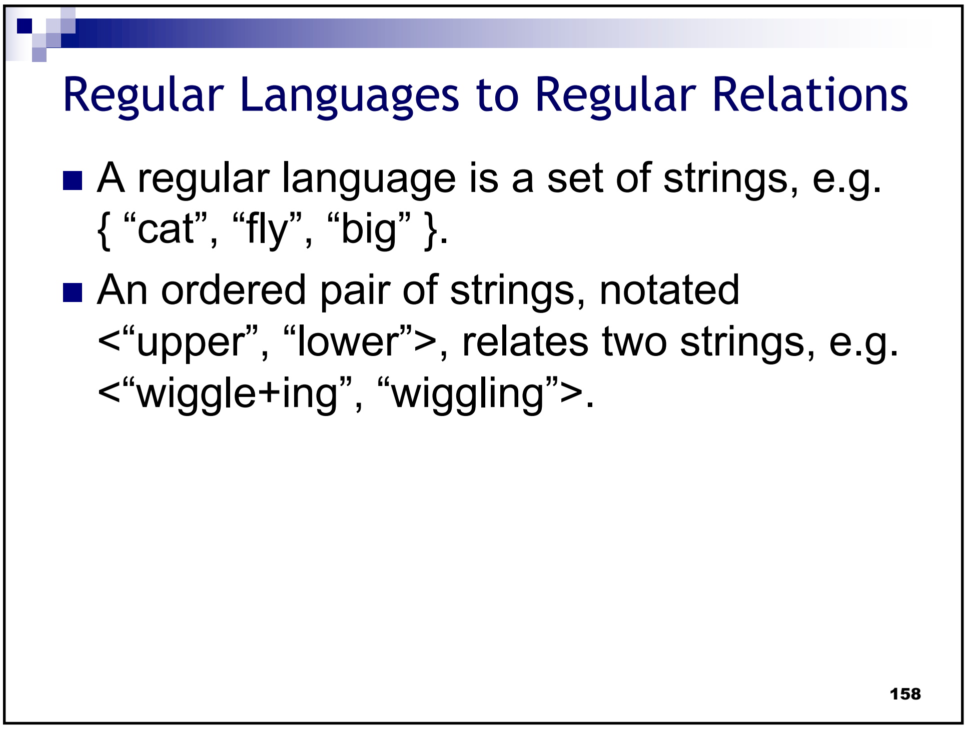
- Assume an alphabet A={a,b,…,z}
- All strings ending in ing
  - A* i n g
- All strings that start with an a and end with a ed
  - a A* e d



Regular Languages to Regular Relations

- A regular language is a set of strings, e.g. { “cat”, “fly”, “big” }.

157



Regular Languages to Regular Relations

- A regular language is a set of strings, e.g. { “cat”, “fly”, “big” }.
- An ordered pair of strings, notated <“upper”, “lower”>, relates two strings, e.g. <“wiggle+ing”, “wiggling”>.

158

Regular Languages to Regular Relations

- A regular language is a set of strings, e.g. { “cat”, “fly”, “big” }.
- An ordered pair of strings, notated <“upper”, “lower”>, relates two strings, e.g. <“wiggle+ing”, “wiggling”>.
- A regular relation is a set of ordered pairs of strings, e.g.
  - { <“cat+N”, “cat”> , <“fly+N”, “fly”> , <“fly+V”, “fly”>, <“big+A”, “big”> }
  - { <“cat”, “cats”> , <“zebra”, “zebras”> , <“deer”, “deer”>, <“ox”, “oxen”>, <“child”, “children”> }

Regular Relations

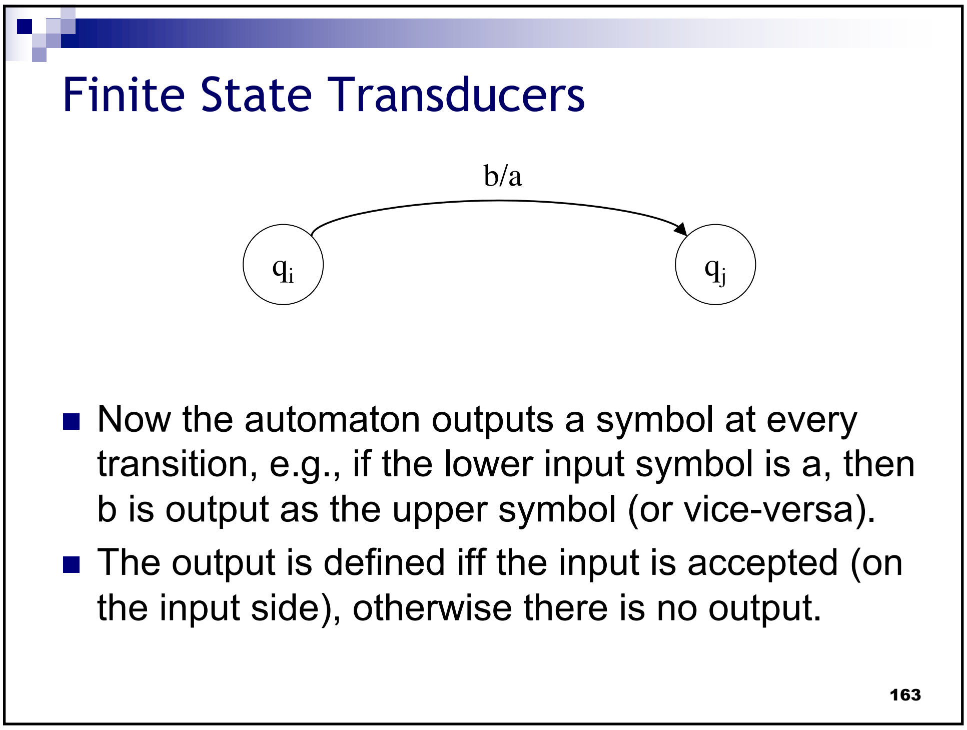
- The set of upper-side strings in a regular relation (upper language) is a regular language.
  - { cat+N , fly+N, fly+V, big+A}
- The set of lower-side strings in a regular relation (lower language) is a regular language.
  - {cat ,fly, big }

Regular Relations

- A regular relation is a “mapping” between two regular ranguages. Each string in one of the languages is “related” to one or more strings of the other language.
- A regular relation can be described in a regular expression and encoded in a Finite-State Transducer (FST).

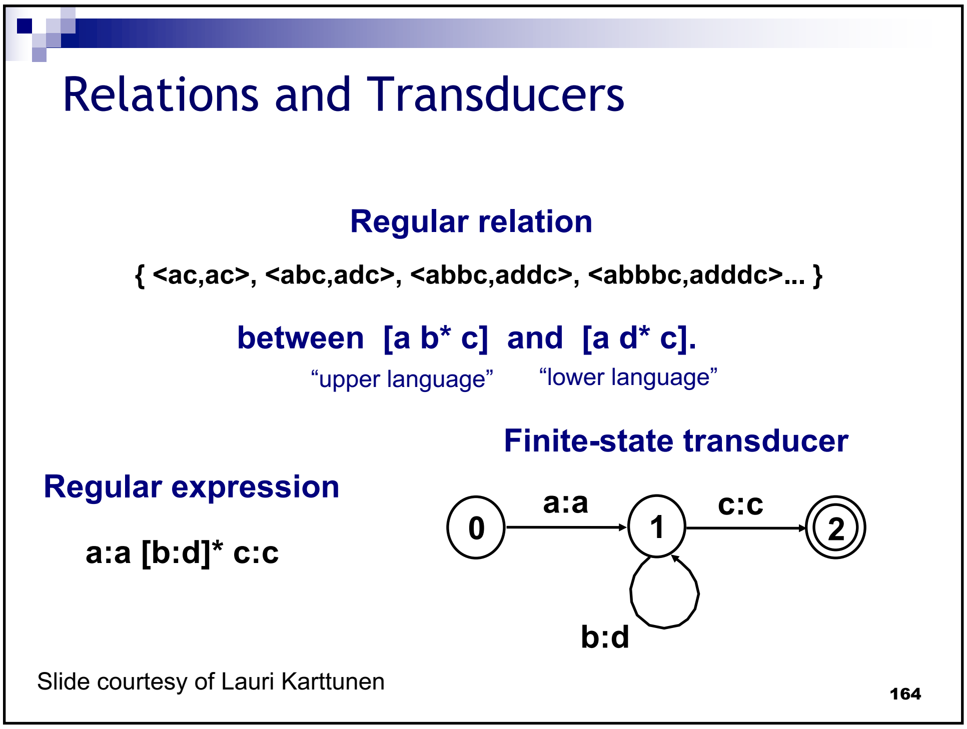
Finite State Transducers

- Regular relations can be defined by regular expressions over an alphabet of pairs of symbols u:l (u for upper, l for lower)
- In general either u or l may be the empty string symbol.
  - □ ε:l: a symbol on the lower side maps to “nothing” on the upper side
  - □ u: ε: a symbol on the upper side maps to “nothing” on the lower side
- It is also convenient to view any recognizer as an identity transducer.



Finite State Transducers

- Now the automaton outputs a symbol at every transition, e.g., if the lower input symbol is a, then b is output as the upper symbol (or vice-versa).
- The output is defined iff the input is accepted (on the input side), otherwise there is no output.



Relations and Transducers

Regular relation

{ <ac,ac>, <abc,adc>, <abbc,addc>, <abbbc,adddc>... }

between [a b* c] and [a d* c].

"upper language" "lower language"

Finite-state transducer

Regular expression

a:a [b:d]* c:c

Slide courtesy of Lauri Karttunen

Relations and Transducers

Regular relation

`{ <ac,ac>, <abc,adc>, <abbc,addc>, <abbbc,adddc>... }`

between [a b* c] and [a d* c].

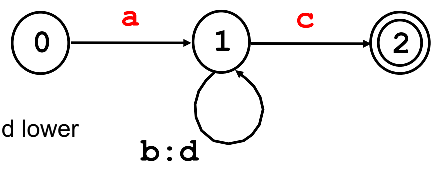
"upper language" "lower language"

Finite-state transducer

Regular expression

`a [b:d]* c`

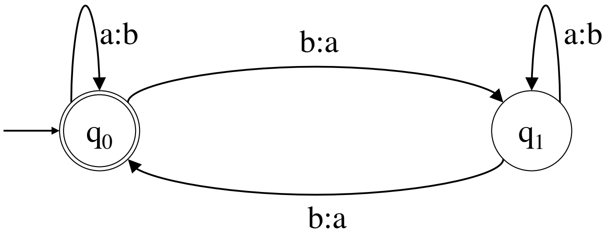
Convention: when both upper and lower symbols are same

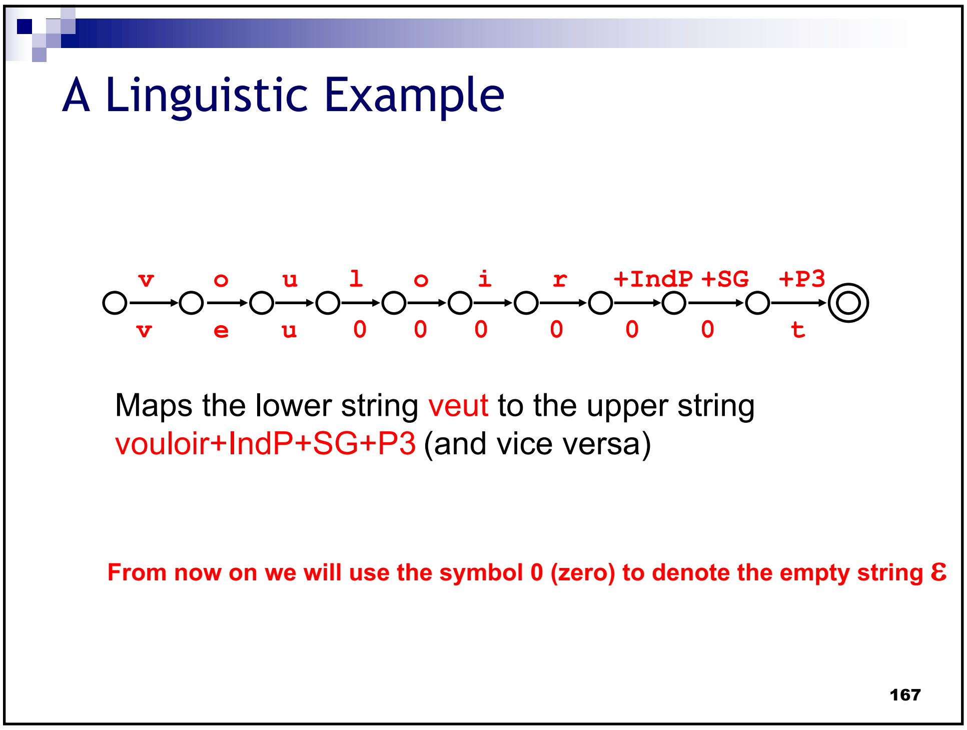


Slide courtesy of Lauri Karttunen

Finite State Transducers

- If the input string has an even number of a's then flip each symbol.

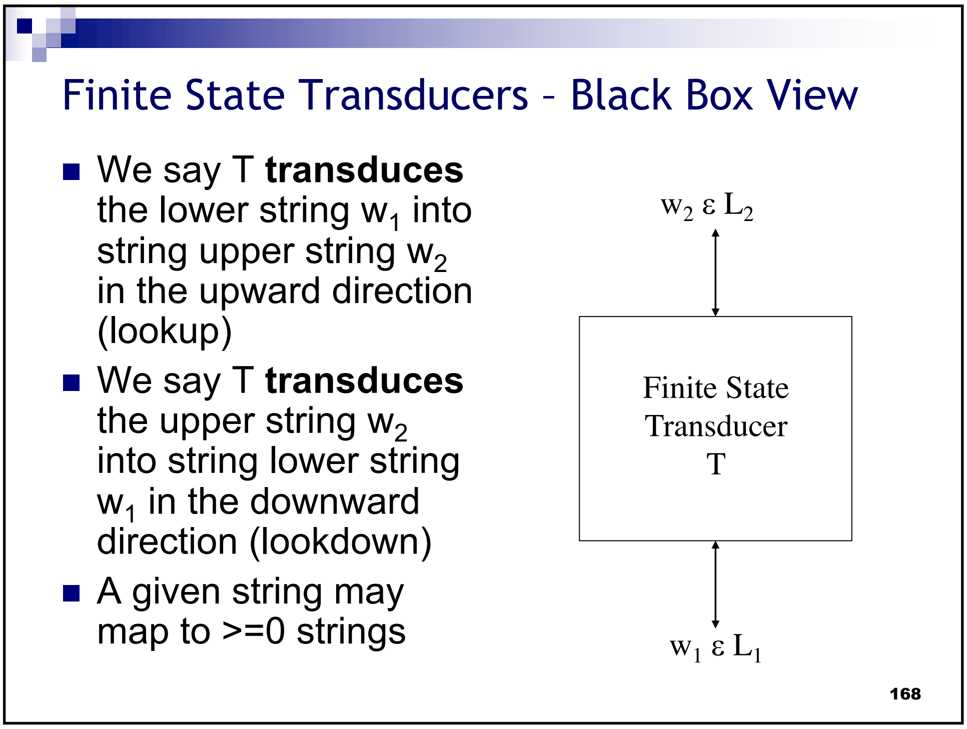




A Linguistic Example

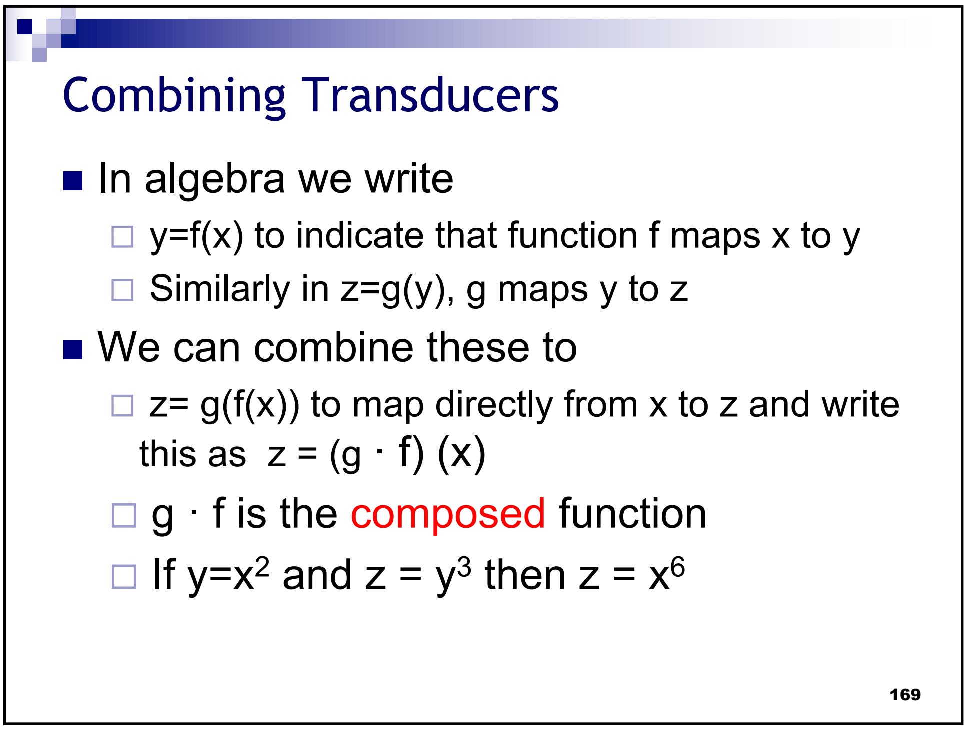
Maps the lower string veut to the upper string vouloir+IndP+SG+P3 (and vice versa)

From now on we will use the symbol 0 (zero) to denote the empty string ε



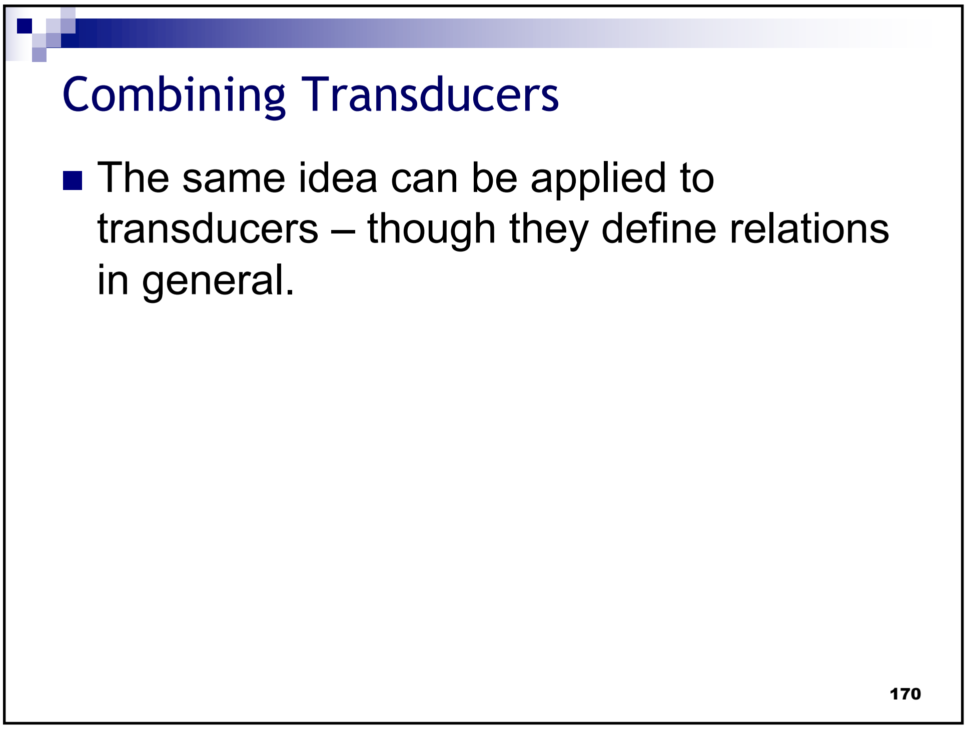
Finite State Transducers - Black Box View

- We say T **transduces** the lower string w_1 into string upper string w_2 in the upward direction (lookup)
- We say T **transduces** the upper string w_2 into string lower string w_1 in the downward direction (lookdown)
- A given string may map to >=0 strings



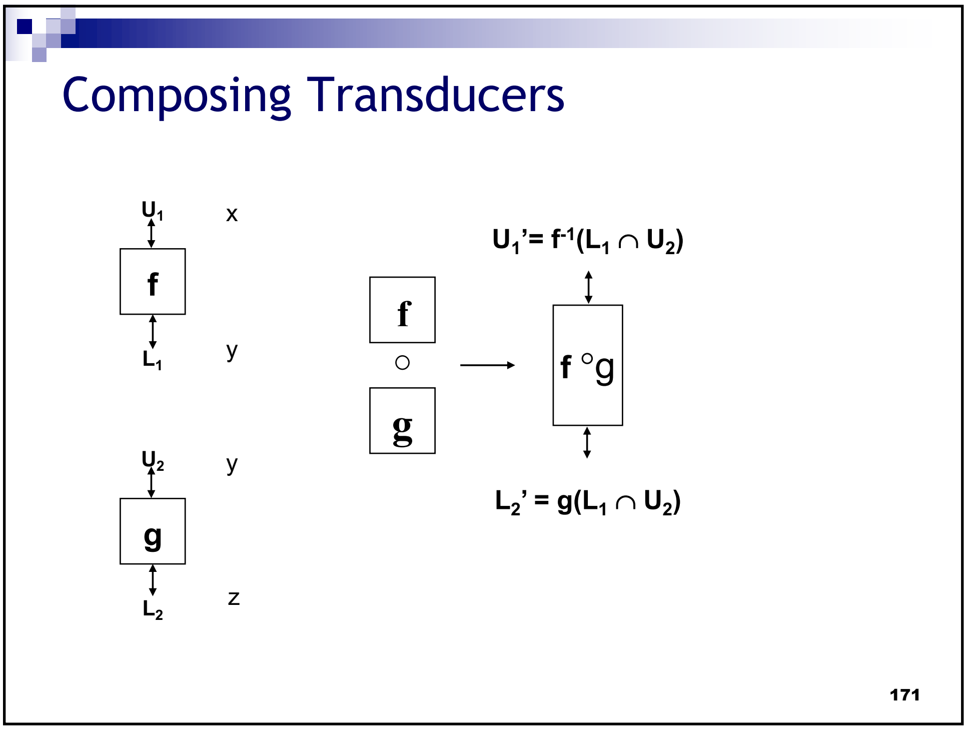
Combining Transducers

- In algebra we write
  - y=f(x) to indicate that function f maps x to y
  - Similarly in z=g(y), g maps y to z
- We can combine these to
  - z= g(f(x)) to map directly from x to z and write this as z = (g · f) (x)
  - g · f is the composed function
  - If $y=x^2$ and $z = y^3$ then $z = x^6$



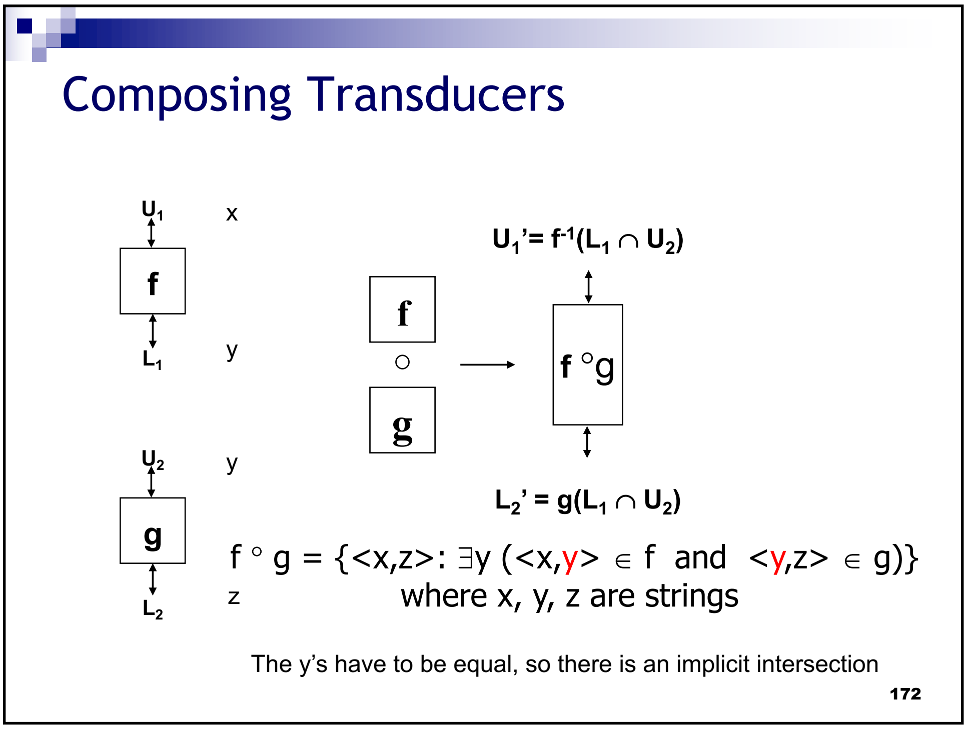
Combining Transducers

- The same idea can be applied to transducers – though they define relations in general.



Composing Transducers

171

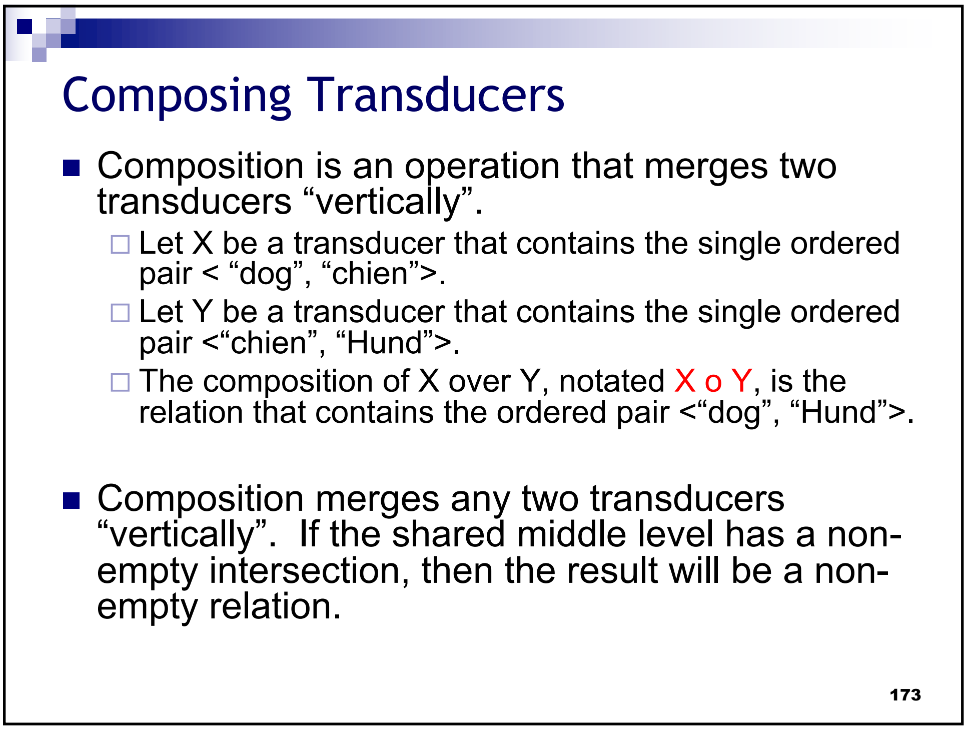


Composing Transducers

$f \circ g = \{<x,z>: \exists y\ (<x,y> \in f \text{ and } <y,z> \in g)\}$
where x, y, z are strings

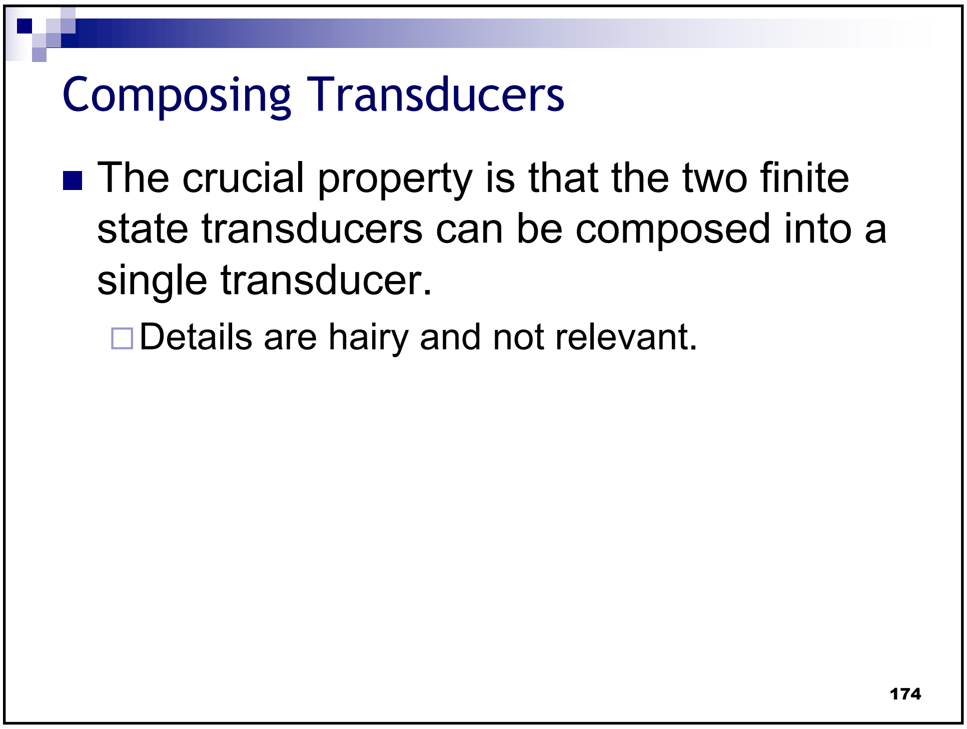
The y's have to be equal, so there is an implicit intersection

172



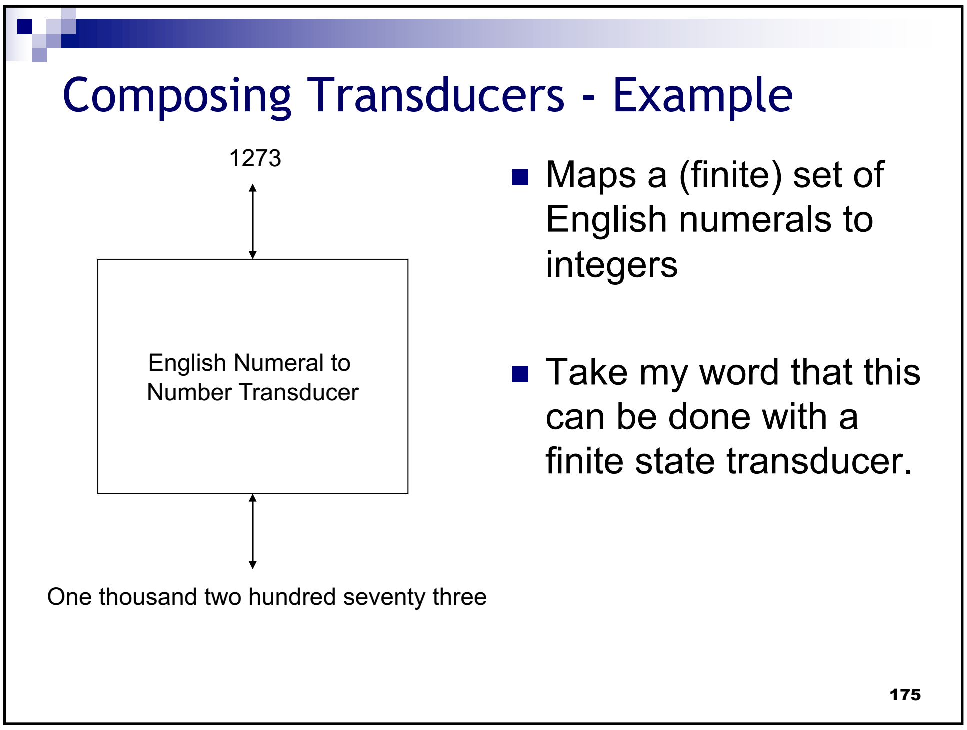
Composing Transducers

- Composition is an operation that merges two transducers “vertically”.
  - □ Let X be a transducer that contains the single ordered pair < “dog”, “chien”>.
  - □ Let Y be a transducer that contains the single ordered pair <“chien”, “Hund”>.
  - □ The composition of X over Y, notated X o Y, is the relation that contains the ordered pair <“dog”, “Hund”>.
- Composition merges any two transducers “vertically”. If the shared middle level has a non-empty intersection, then the result will be a non-empty relation.



Composing Transducers

- The crucial property is that the two finite state transducers can be composed into a single transducer.
  - □ Details are hairy and not relevant.



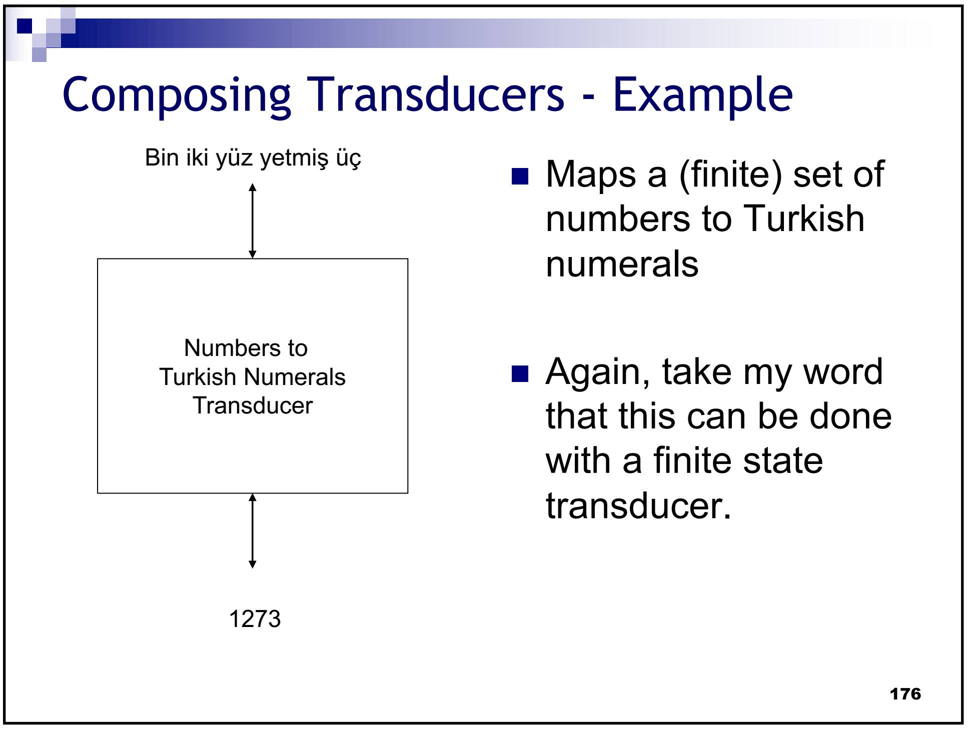
Composing Transducers - Example

1273

English Numeral to Number Transducer

One thousand two hundred seventy three

- Maps a (finite) set of English numerals to integers
- Take my word that this can be done with a finite state transducer.



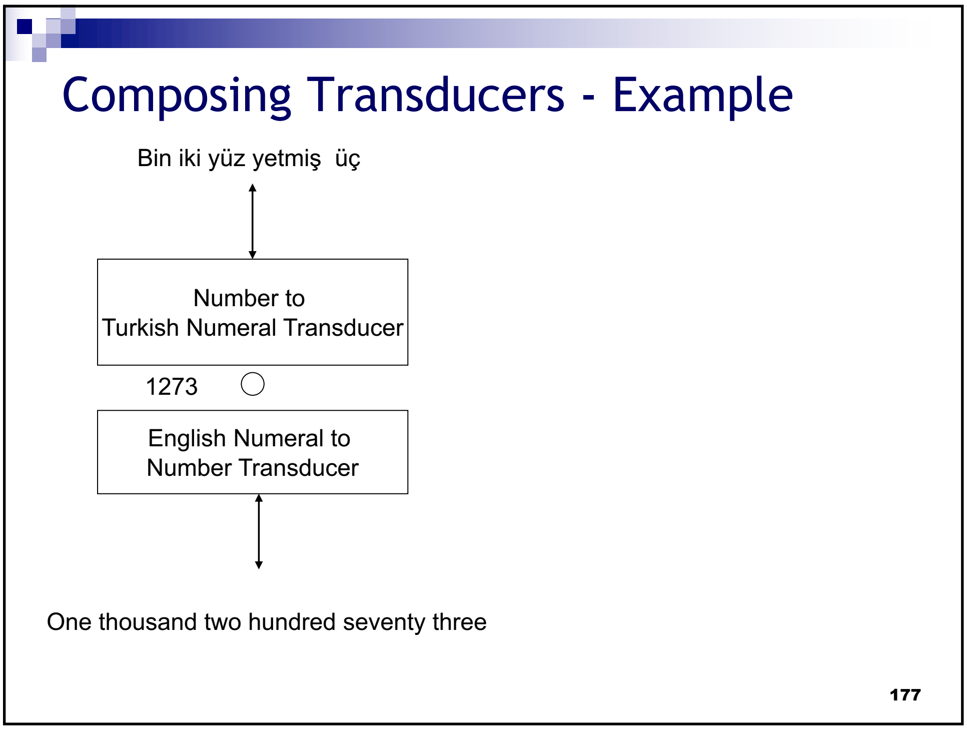
Composing Transducers - Example

Bin iki yüz yetmiş üç

Numbers to Turkish Numerals Transducer

1273

- Maps a (finite) set of numbers to Turkish numerals
- Again, take my word that this can be done with a finite state transducer.



Composing Transducers - Example

Bin iki yüz yetmiş üç

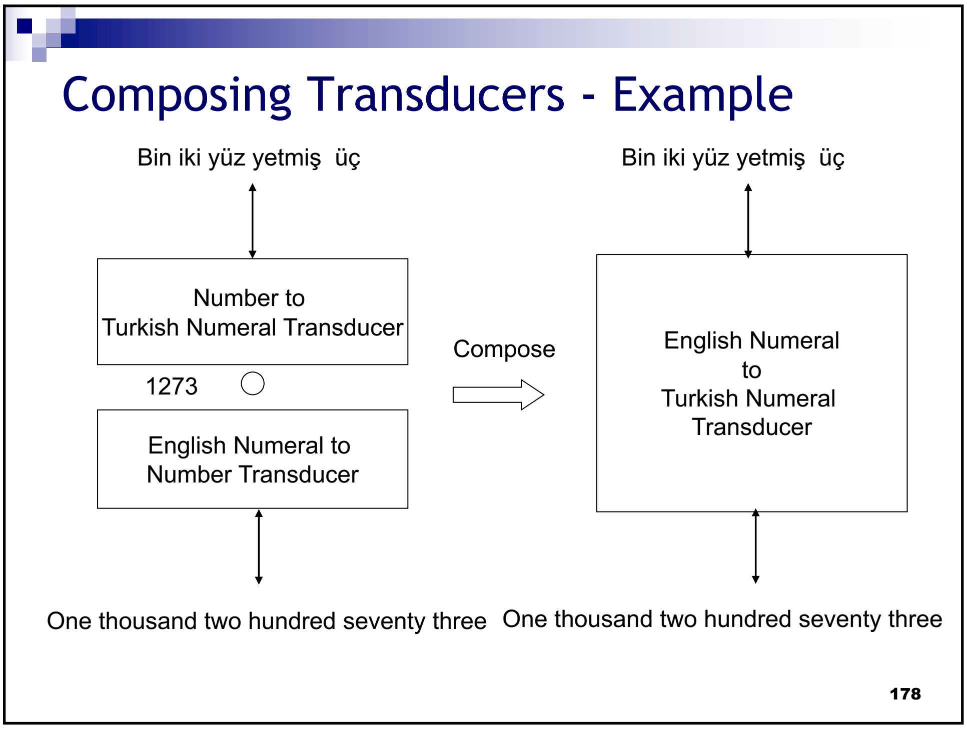
Number to
Turkish Numeral Transducer

1273 ○

English Numeral to
Number Transducer

One thousand two hundred seventy three

177



Composing Transducers - Example

Bin iki yüz yetmiş üç

Number to
Turkish Numeral Transducer

1273 ○

English Numeral to
Number Transducer

One thousand two hundred seventy three

Compose

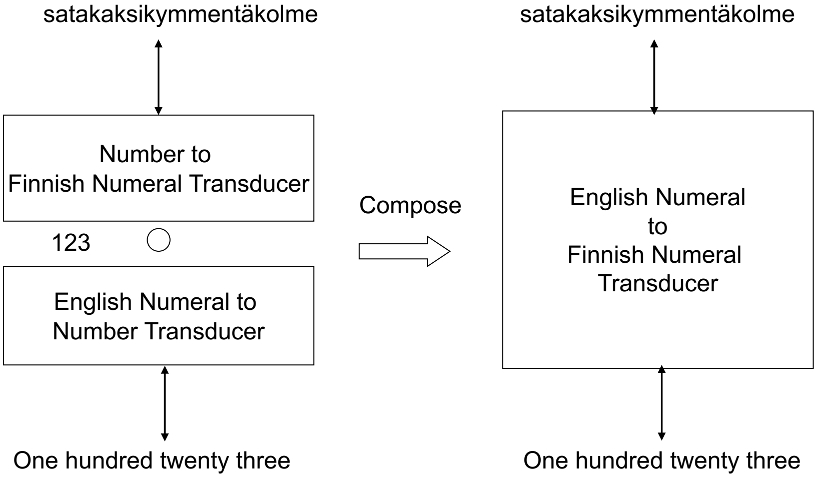
Bin iki yüz yetmiş üç

English Numeral
to
Turkish Numeral
Transducer

One thousand two hundred seventy three

178

Composing Transducers - Example



satakaksikymmentäkolme

Number to Finnish Numeral Transducer

123 ○

English Numeral to Number Transducer

One hundred twenty three

Compose

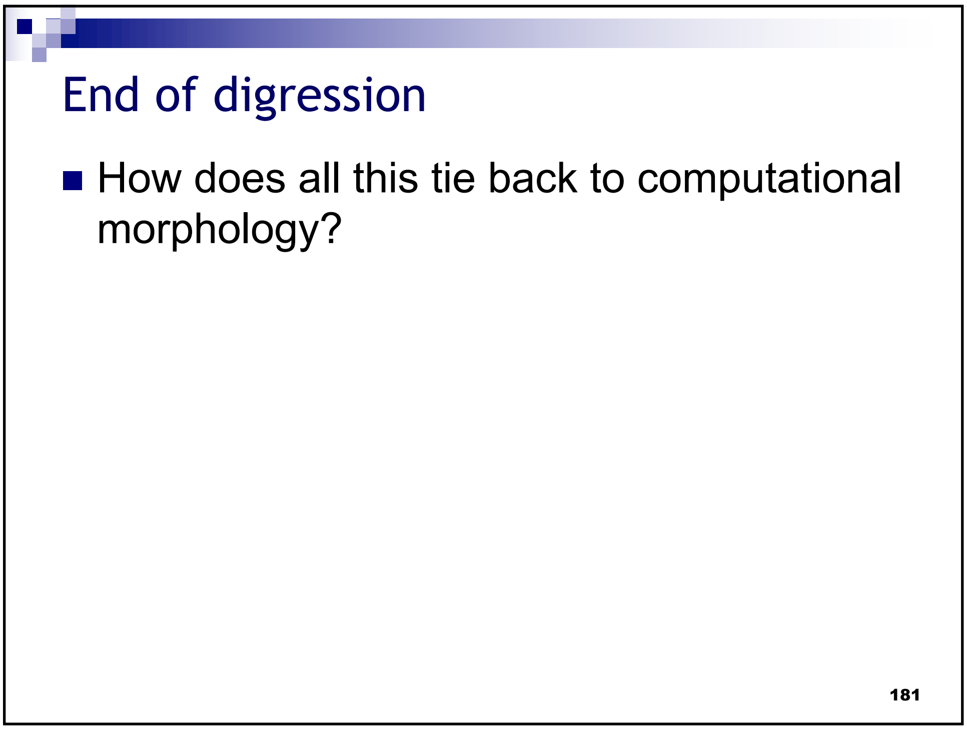
satakaksikymmentäkolme

English Numeral to Finnish Numeral Transducer

One hundred twenty three

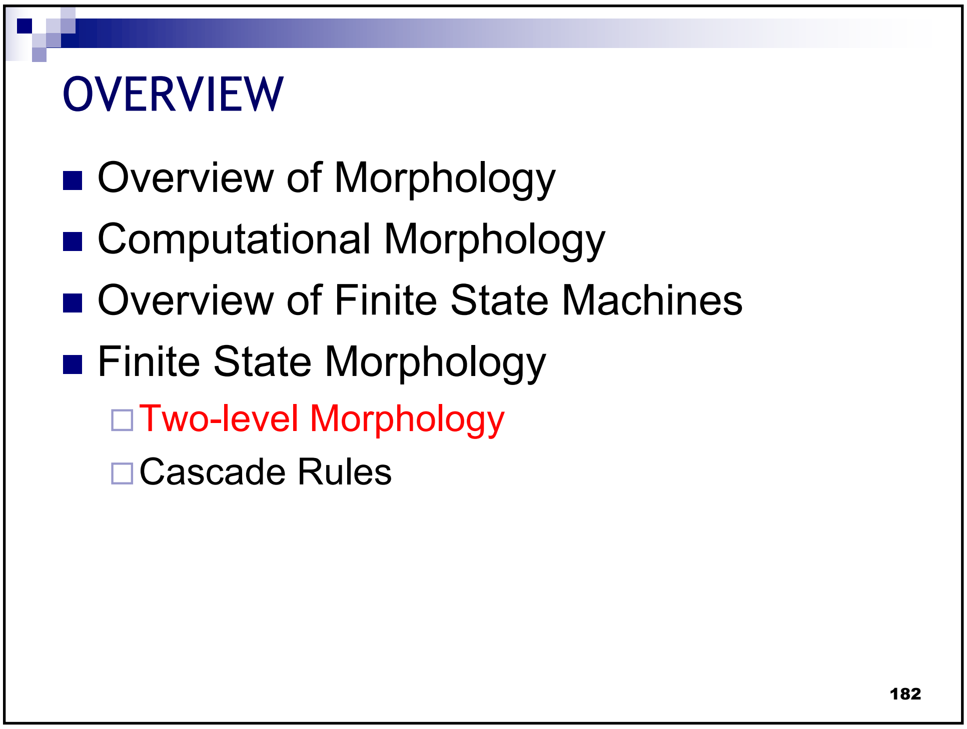
Morphology & Finite State Concepts

- Some references:
  - □ Hopcroft, Motwani and Ullman, Formal Languages and Automata Theory, Addison Wesley
  - □ Beesley and Karttunen, Finite State Morphology, CSLI, 2003
  - □ Kaplan and Kay. “Regular Models of Phonological Rule Systems” in *Computational Linguistics* 20:3, 1994. Pp. 331-378.
  - □ Karttunen et al., Regular Expressions for Language Engineering, *Natural Language Engineering*, 1996



End of digression

- How does all this tie back to computational morphology?



OVERVIEW

- Overview of Morphology
- Computational Morphology
- Overview of Finite State Machines
- Finite State Morphology
  - □ Two-level Morphology
  - □ Cascade Rules

Morphological Analysis and Regular Sets

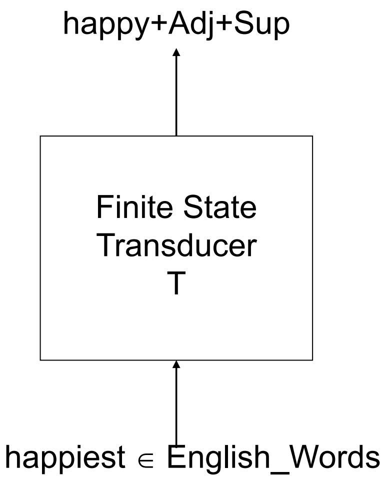
- Assumption:

  The set of words in a (natural) language is a regular (formal) language.

  - Finite: Trivially true; though instead of listing all words, one needs to abstract and describe phenomena.
  - Infinite: Very accurate approximation.
- BUT: Processes like (unbounded) reduplication are out of the finite state realm.
- Let's also assume that we combine morphemes by simple concatenation.
  - Not a serious limitation

Morphological Analysis

- Morphological analysis can be seen as a finite state transduction



Morphological Analysis as FS Transduction

- First approximation
- Need to describe
  - □ Lexicon (of free and bound morphemes)
  - □ Spelling change rules in a finite state framework.

The Lexicon as a Finite State Transducer

- Assume words have the form prefix+root+suffix where the prefix and the suffix are optional.

  So:

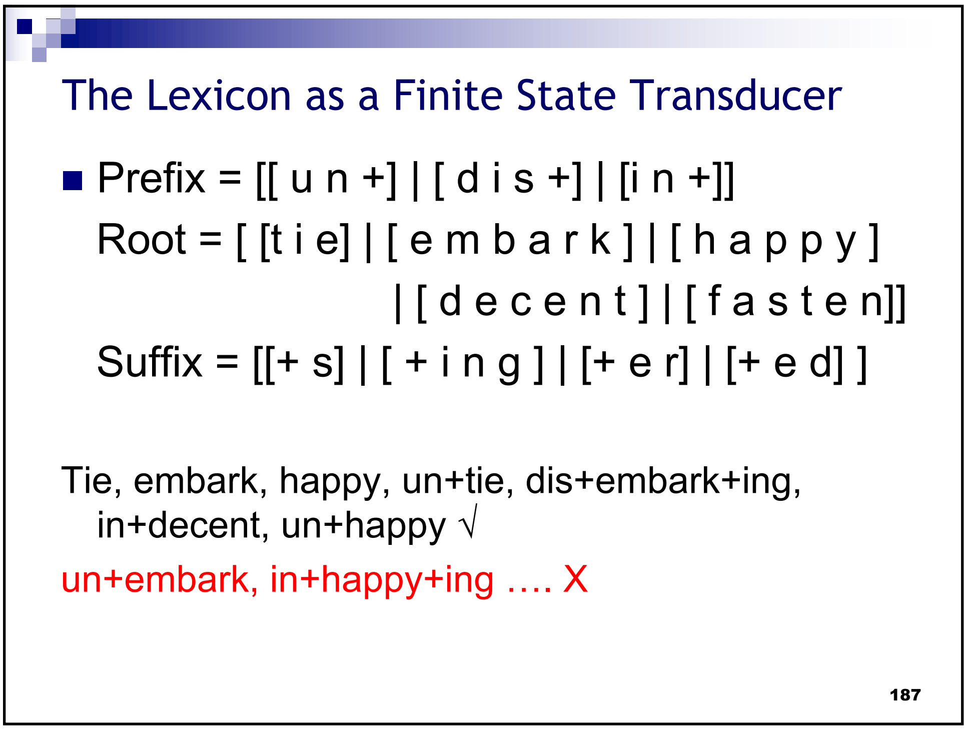
  Prefix = $[P_1 \mid P_2 \mid \ldots \mid P_k]$

  Root = $[R_1 \mid R_2 \mid \ldots \mid R_m]$

  Suffix = $[S_1 \mid S_2 \mid \ldots \mid S_n]$

  Lexicon = (Prefix) Root (Suffix)

(R) = [R | ε], that is, R is optional.

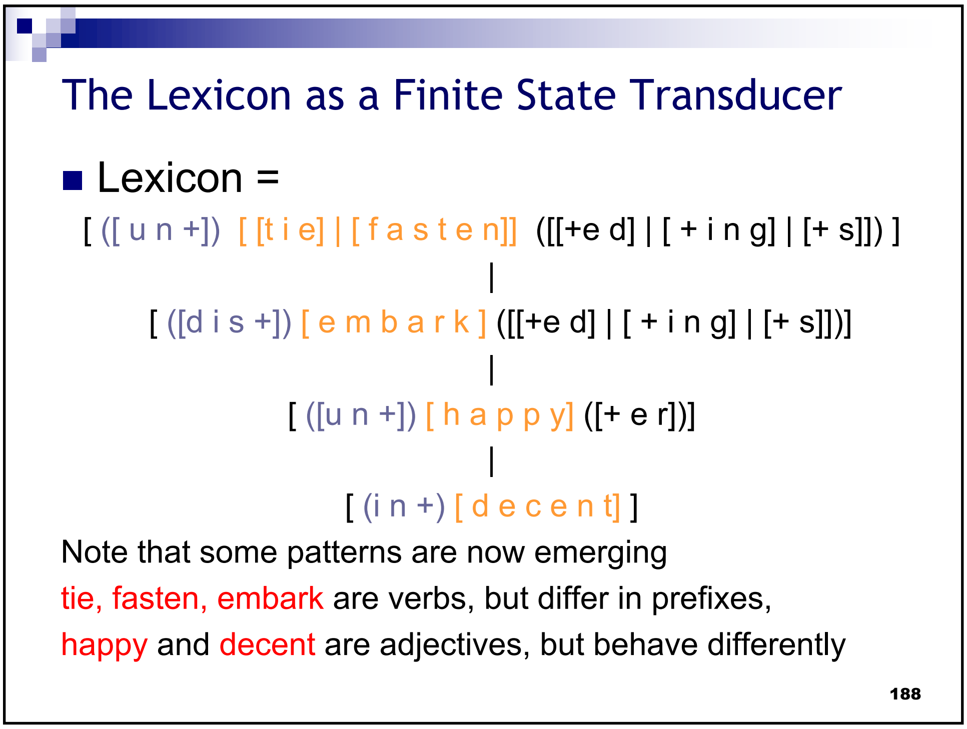


The Lexicon as a Finite State Transducer

- Prefix = [[u n +] | [d i s +] | [i n +]]
  Root = [[t i e] | [e m b a r k] | [h a p p y]
  | [d e c e n t] | [f a s t e n]]
  Suffix = [[+ s] | [+ i n g] | [+ e r] | [+ e d]]

Tie, embark, happy, un+tie, dis+embark+ing, in+decent, un+happy √

un+embark, in+happy+ing …. X



The Lexicon as a Finite State Transducer

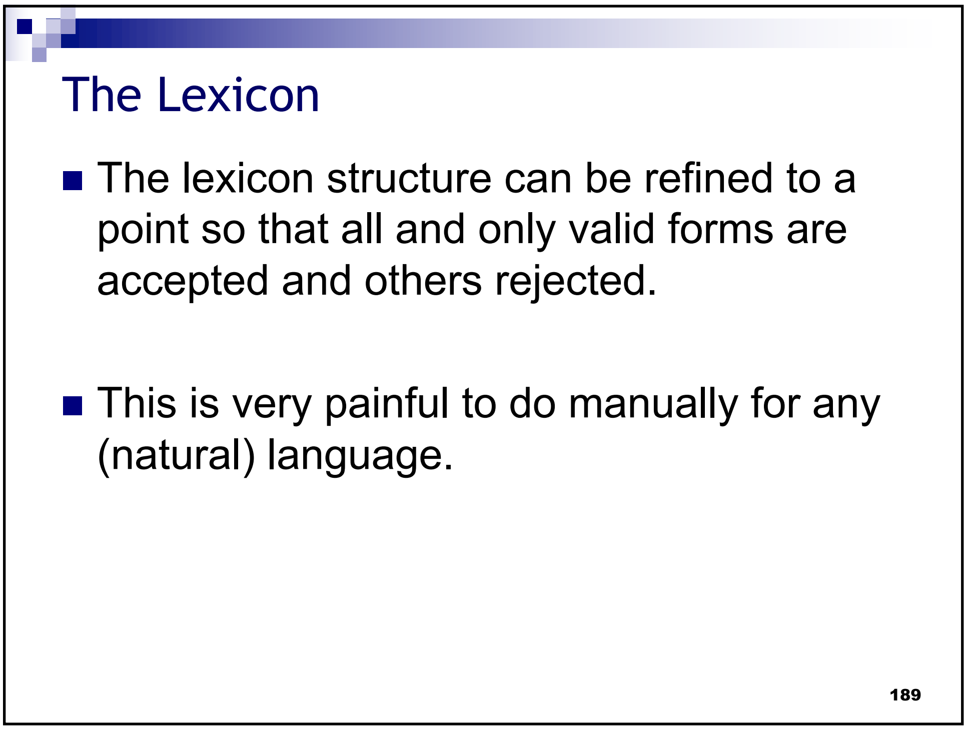
- Lexicon =

```
[ ([ u n +])  [ [t i e] | [ f a s t e n]]  ([[+e d] | [ + i n g] | [+ s]]) ]
                                    |
        [ ([d i s +]) [ e m b a r k ] ([[+e d] | [ + i n g] | [+ s]])]
                                    |
                   [ ([u n +]) [ h a p p y] ([+ e r])]
                                    |
                         [ (i n +) [ d e c e n t] ]
```

Note that some patterns are now emerging

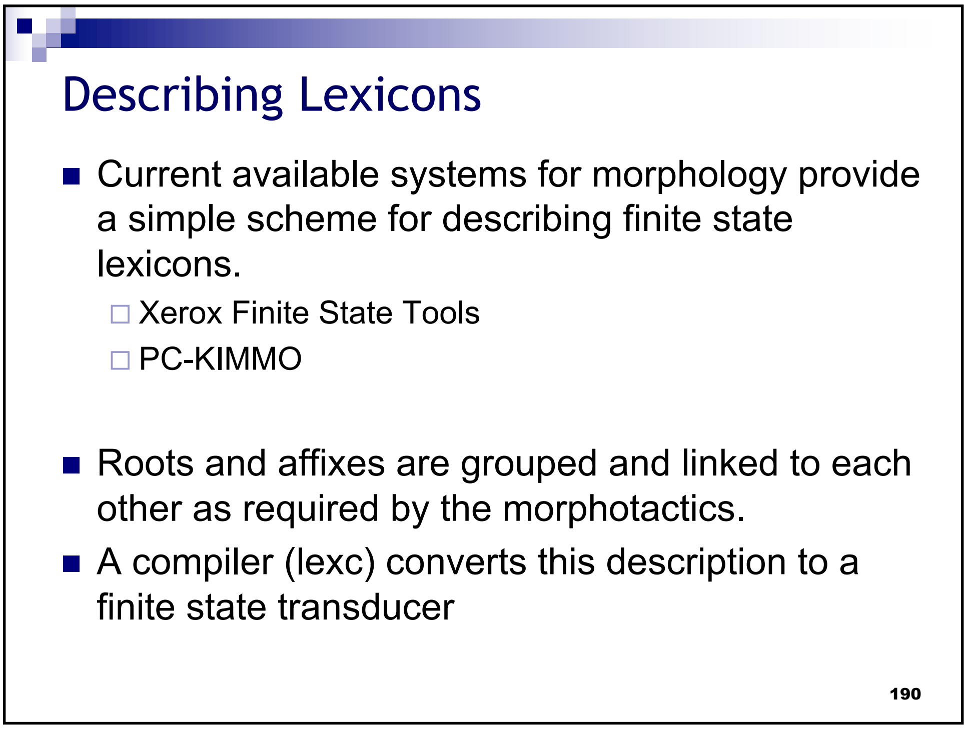
tie, fasten, embark are verbs, but differ in prefixes,

happy and decent are adjectives, but behave differently



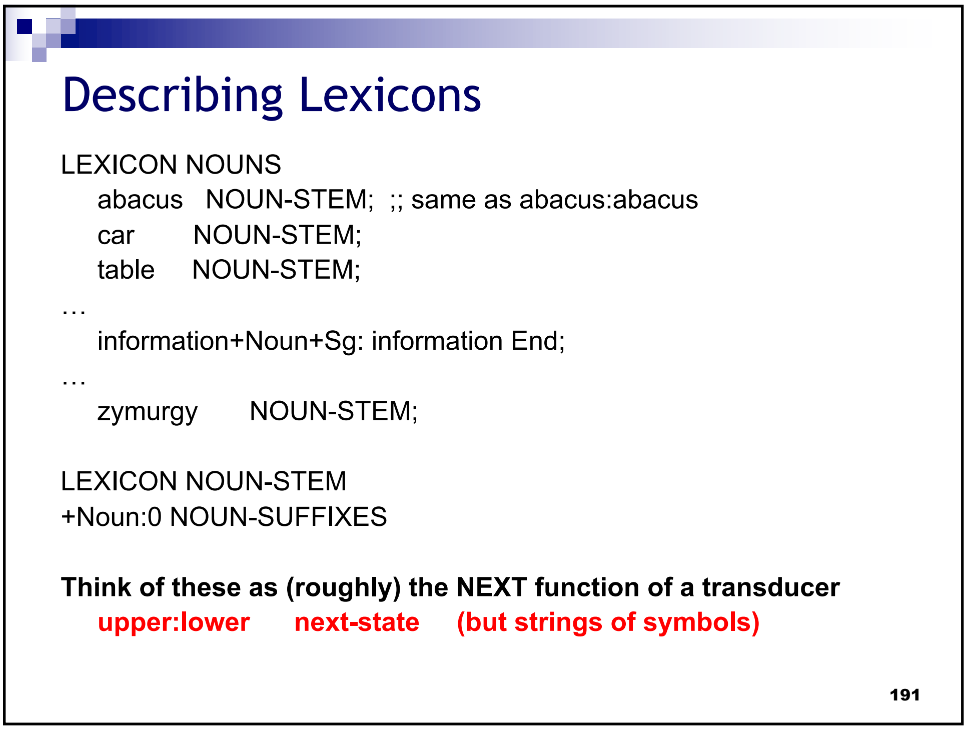
The Lexicon

- The lexicon structure can be refined to a point so that all and only valid forms are accepted and others rejected.
- This is very painful to do manually for any (natural) language.



Describing Lexicons

- Current available systems for morphology provide a simple scheme for describing finite state lexicons.
  - □ Xerox Finite State Tools
  - □ PC-KIMMO
- Roots and affixes are grouped and linked to each other as required by the morphotactics.
- A compiler (lexc) converts this description to a finite state transducer



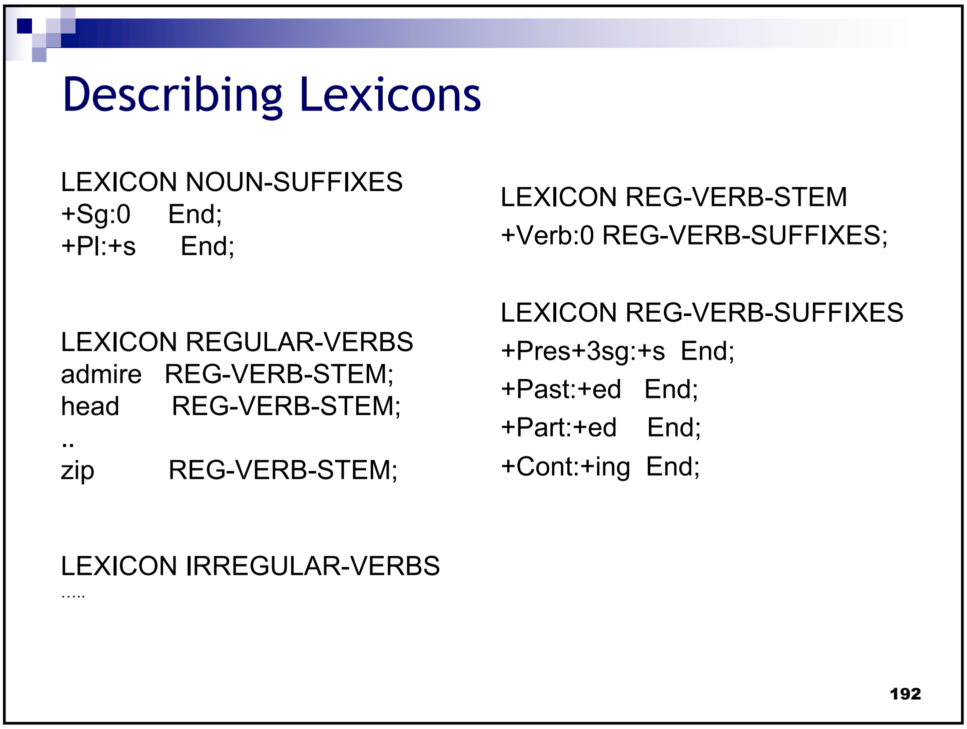
Describing Lexicons

```
LEXICON NOUNS
    abacus   NOUN-STEM;  ;; same as abacus:abacus
    car       NOUN-STEM;
    table     NOUN-STEM;
…
    information+Noun+Sg: information End;
…
    zymurgy       NOUN-STEM;

LEXICON NOUN-STEM
+Noun:0 NOUN-SUFFIXES
```

Think of these as (roughly) the NEXT function of a transducer

upper:lower   next-state   (but strings of symbols)

191



Describing Lexicons

```
LEXICON NOUN-SUFFIXES
+Sg:0     End;
+Pl:+s     End;


LEXICON REGULAR-VERBS
admire   REG-VERB-STEM;
head       REG-VERB-STEM;
..
zip         REG-VERB-STEM;


LEXICON IRREGULAR-VERBS
.....
```

```
LEXICON REG-VERB-STEM
+Verb:0 REG-VERB-SUFFIXES;

LEXICON REG-VERB-SUFFIXES
+Pres+3sg:+s  End;
+Past:+ed   End;
+Part:+ed    End;
+Cont:+ing  End;
```

192

How Does it Work?

- Suppose we have the string abacus+s
  - The first segment matches abacus in the NOUNS lexicon, “outputs” abacus and we next visit the NOUN-STEM lexicon.
  - Here, the only option is to match to the empty string and “output” +Noun and visit the NOUN-SUFFIXES lexicon.
  - Here we have two options
    - Match the epsilon and “output” +Sg but there is more stuff so this fails.
    - Match the +s and “output” +Plu and the finish.
  - Output is abacus+Noun+Pl

Describing Lexicons

LEXICON ADJECTIVES

…

LEXICON ADVERBS

…

LEXICON ROOT

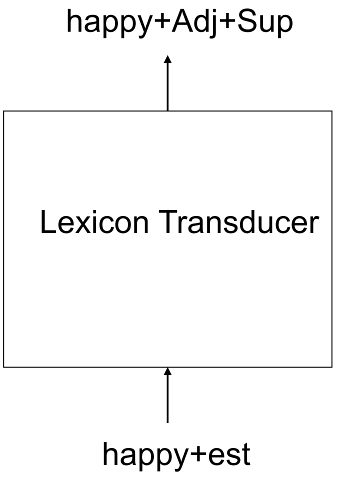
NOUNS;

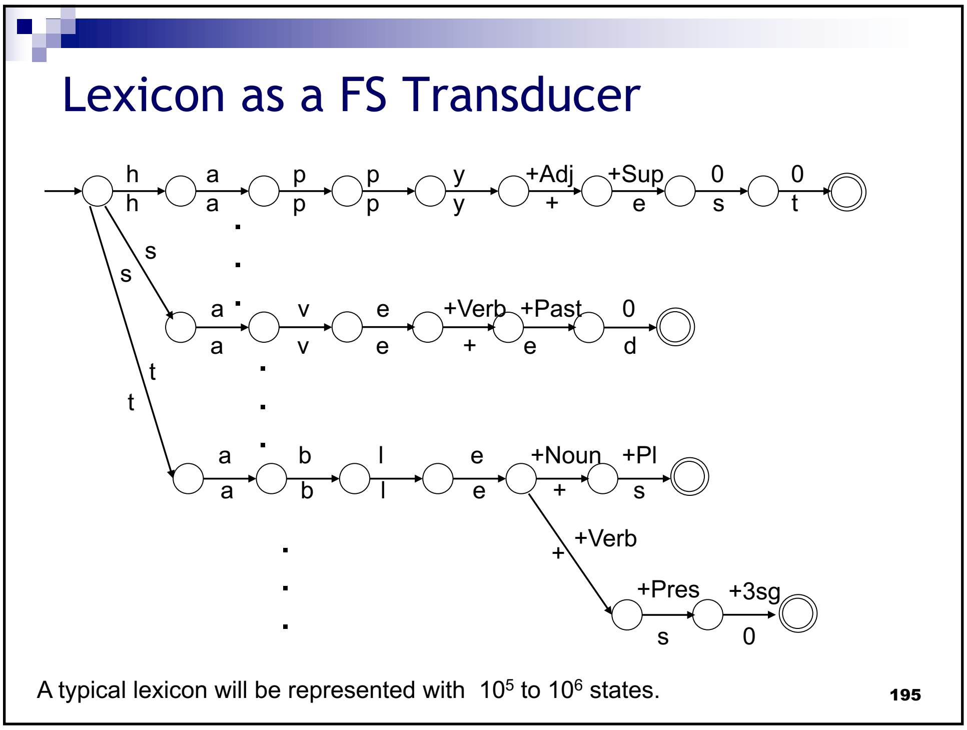
REGULAR-VERBS;

….

ADJECTIVES;

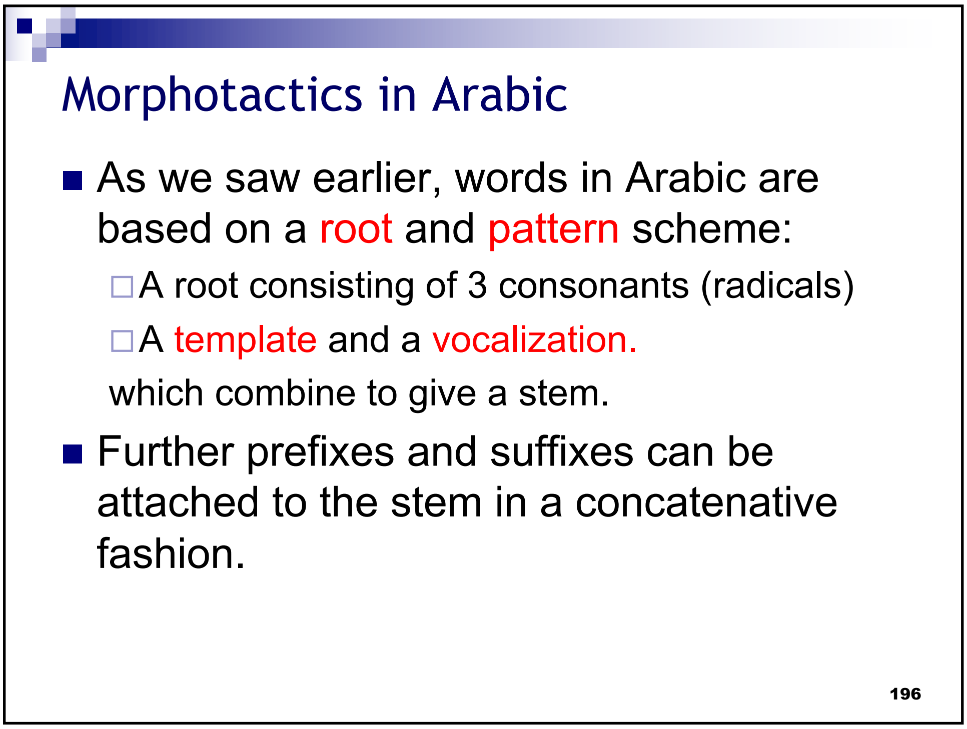
….





Lexicon as a FS Transducer

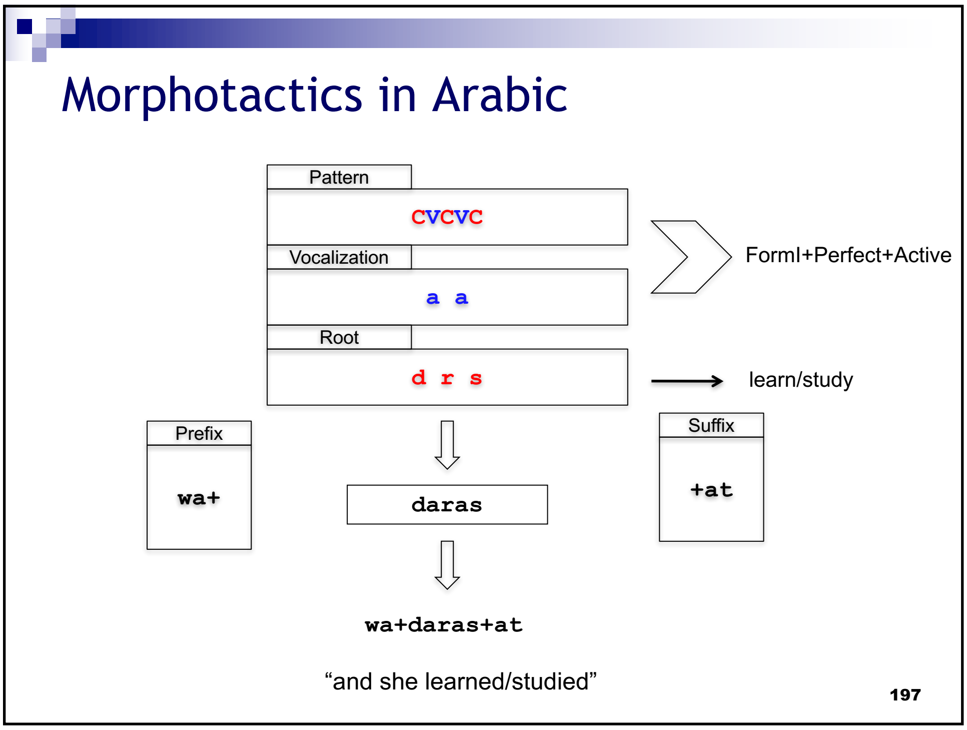
A typical lexicon will be represented with 10^5 to 10^6 states.



Morphotactics in Arabic

- As we saw earlier, words in Arabic are based on a root and pattern scheme:
  - □ A root consisting of 3 consonants (radicals)
  - □ A template and a vocalization.

  which combine to give a stem.
- Further prefixes and suffixes can be attached to the stem in a concatenative fashion.



Morphotactics in Arabic

Pattern

CVCVC

Vocalization

a a

Root

d r s

FormI+Perfect+Active

learn/study

Prefix

wa+

daras

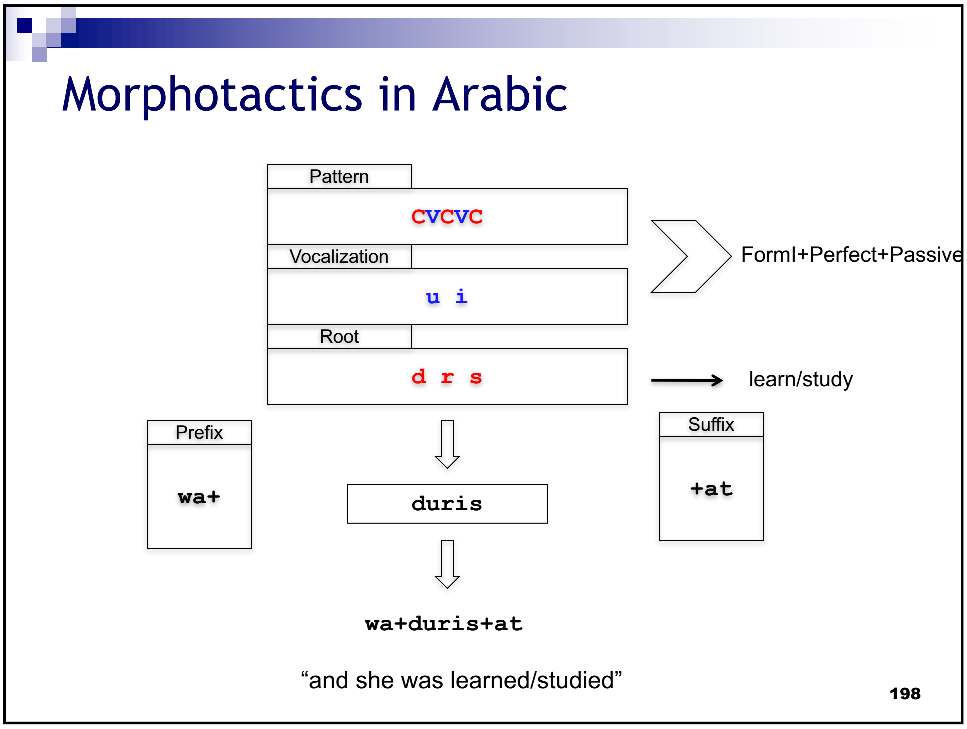
Suffix

+at

wa+daras+at

“and she learned/studied”

197



Morphotactics in Arabic

Pattern

CVCVC

Vocalization

u i

Root

d r s

FormI+Perfect+Passive

learn/study

Prefix

wa+

duris

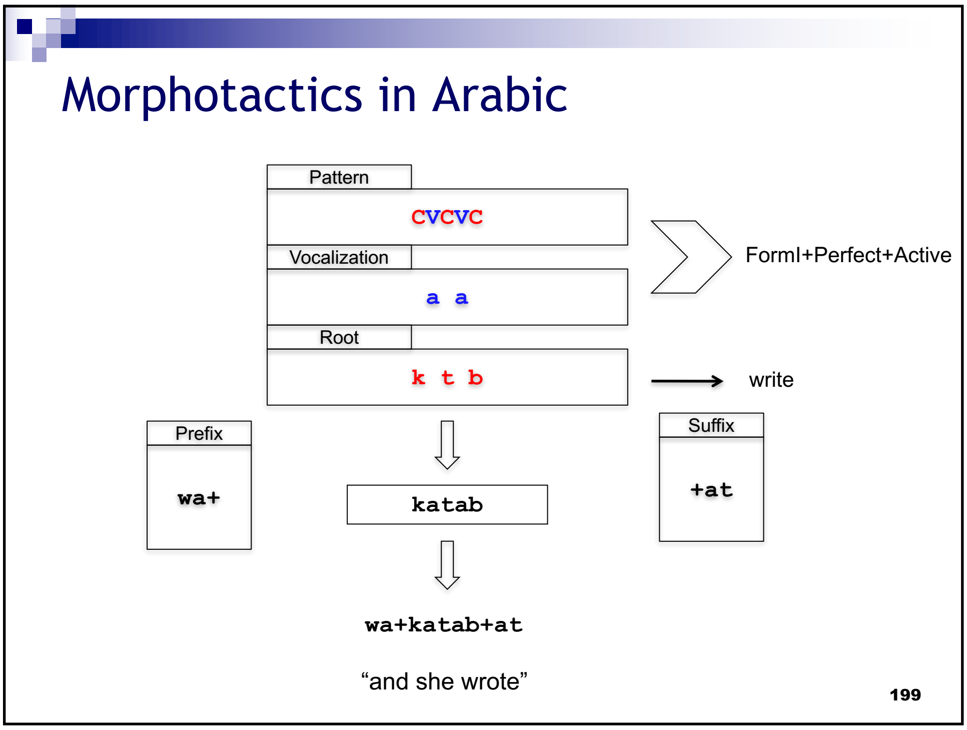
Suffix

+at

wa+duris+at

“and she was learned/studied”

198



Morphotactics in Arabic

Pattern

CVCVC

Vocalization

a a

Root

k t b

FormI+Perfect+Active

write

Prefix

wa+

katab

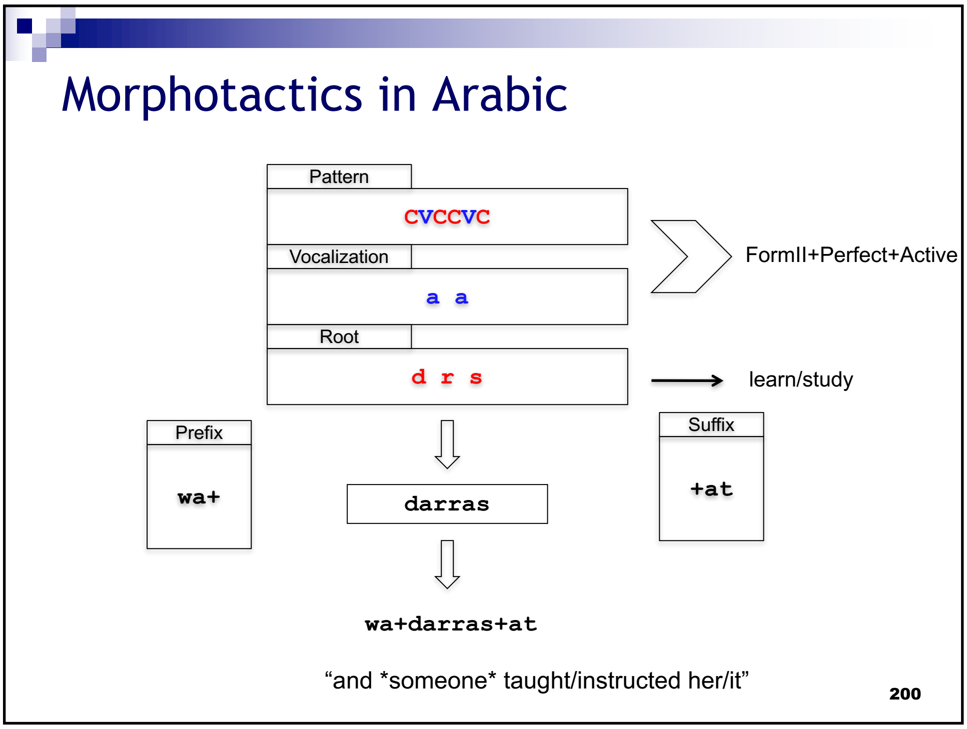
Suffix

+at

wa+katab+at

“and she wrote”

199



Morphotactics in Arabic

Pattern

CVCCVC

Vocalization

a a

Root

d r s

FormII+Perfect+Active

learn/study

Prefix

wa+

darras

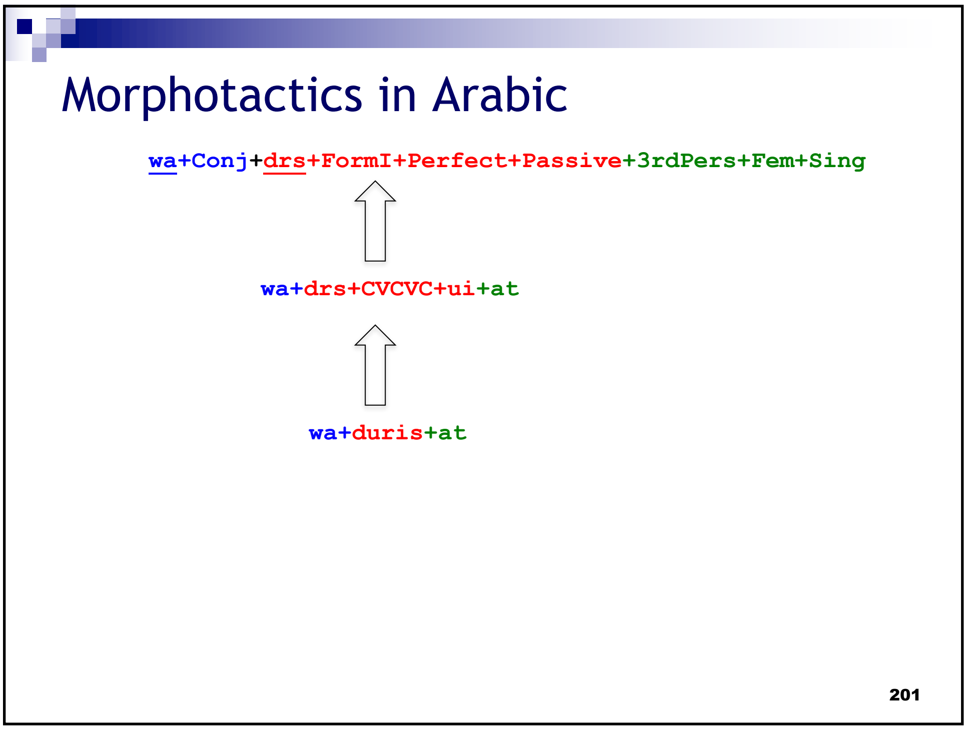
Suffix

+at

wa+darras+at

“and *someone* taught/instructed her/it”

200



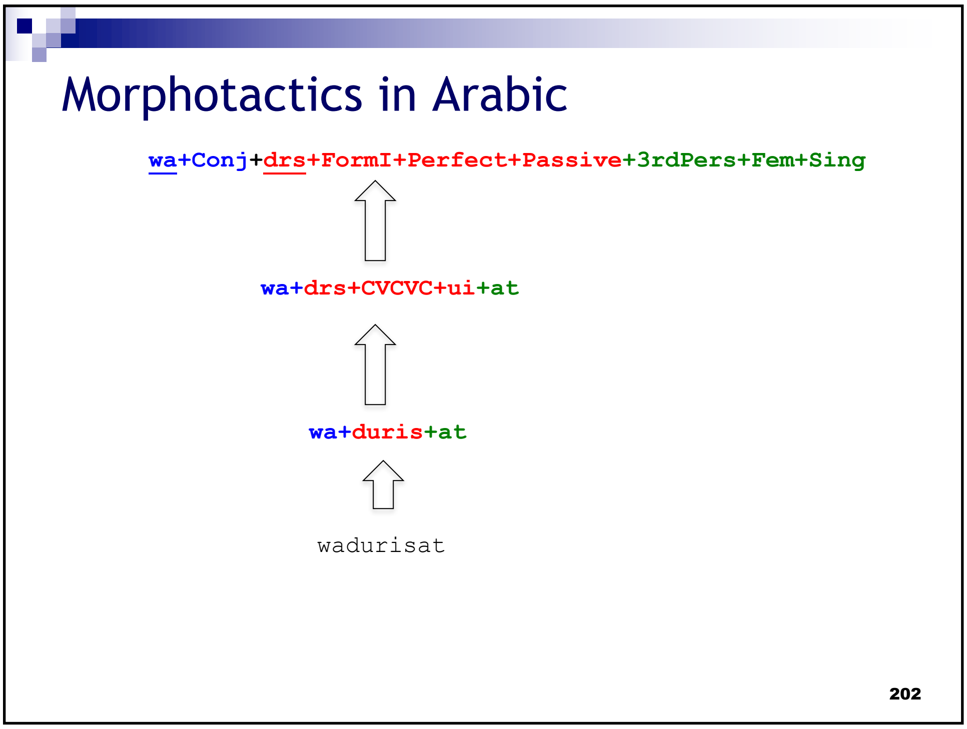
Morphotactics in Arabic

wa+Conj+drs+FormI+Perfect+Passive+3rdPers+Fem+Sing

wa+drs+CVCVC+ui+at

wa+duris+at

201



Morphotactics in Arabic

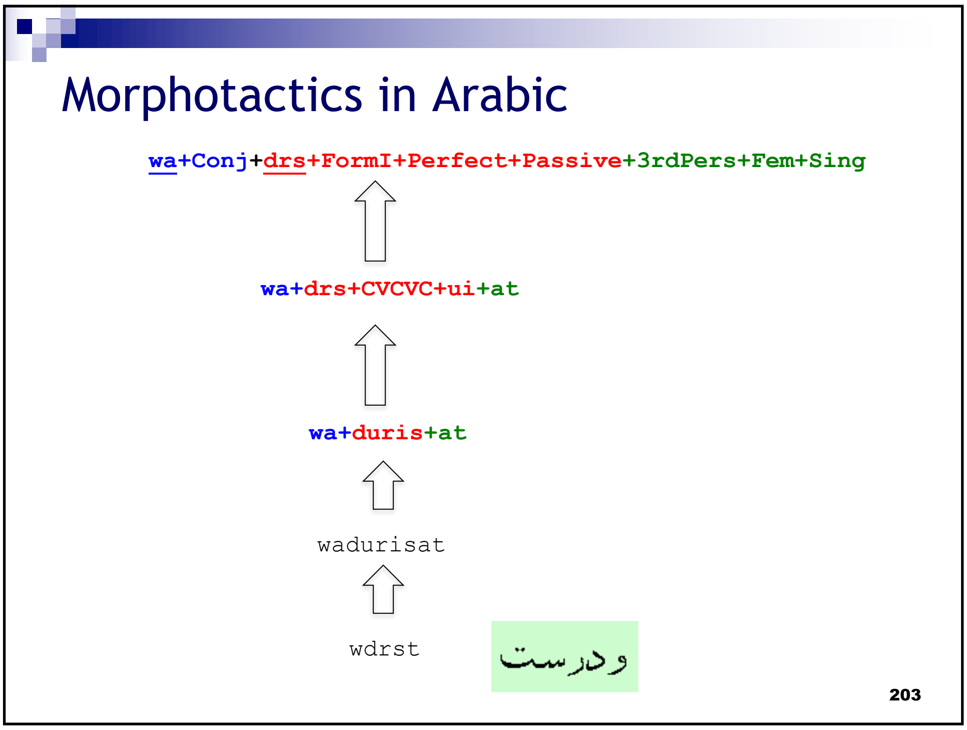
wa+Conj+drs+FormI+Perfect+Passive+3rdPers+Fem+Sing

wa+drs+CVCVC+ui+at

wa+duris+at

wadurisat

202



Morphotactics in Arabic

wa+Conj+drs+FormI+Perfect+Passive+3rdPers+Fem+Sing

wa+drs+CVCVC+ui+at

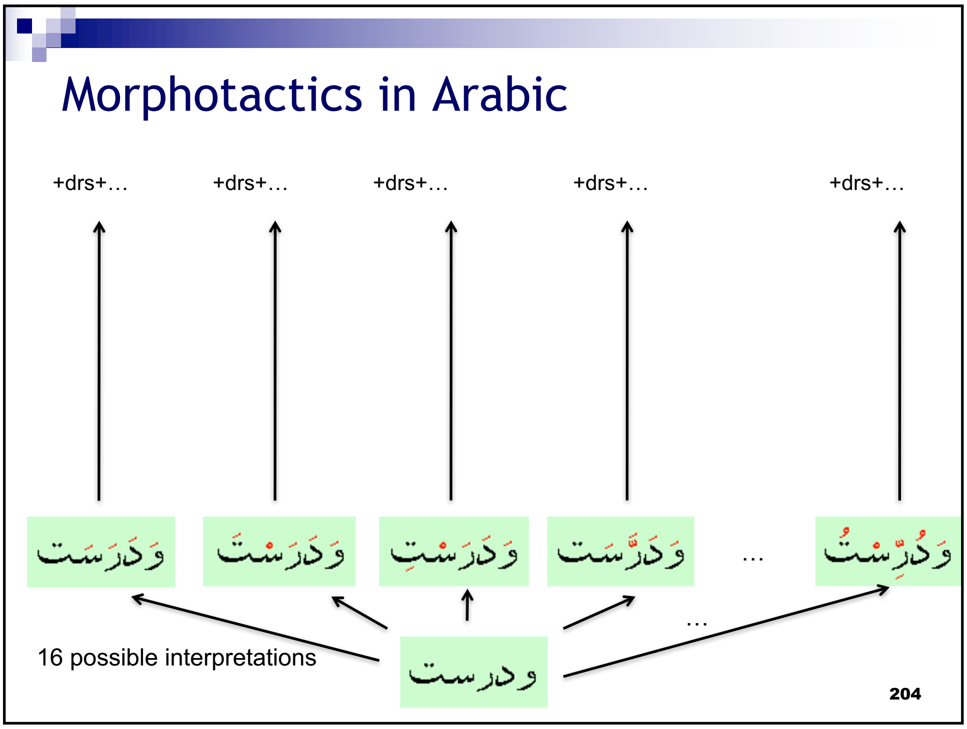
wa+duris+at

wadurisat

wdrst

ودرست

203



Morphotactics in Arabic

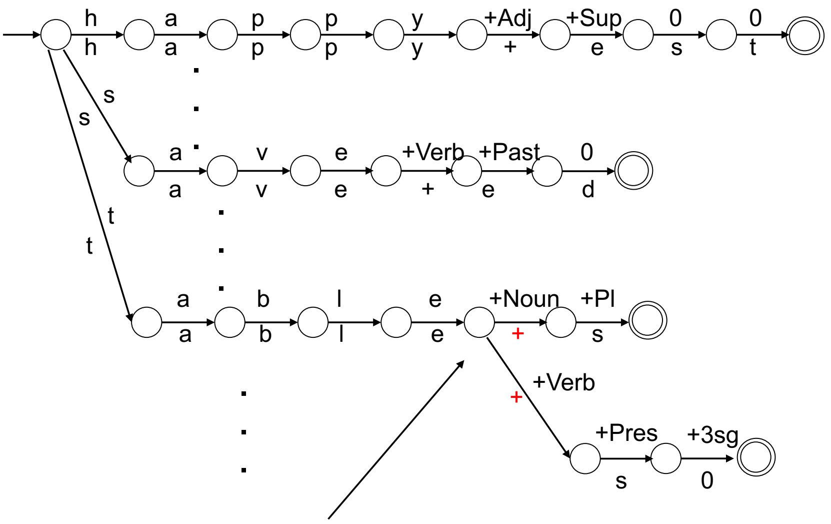
16 possible interpretations

204

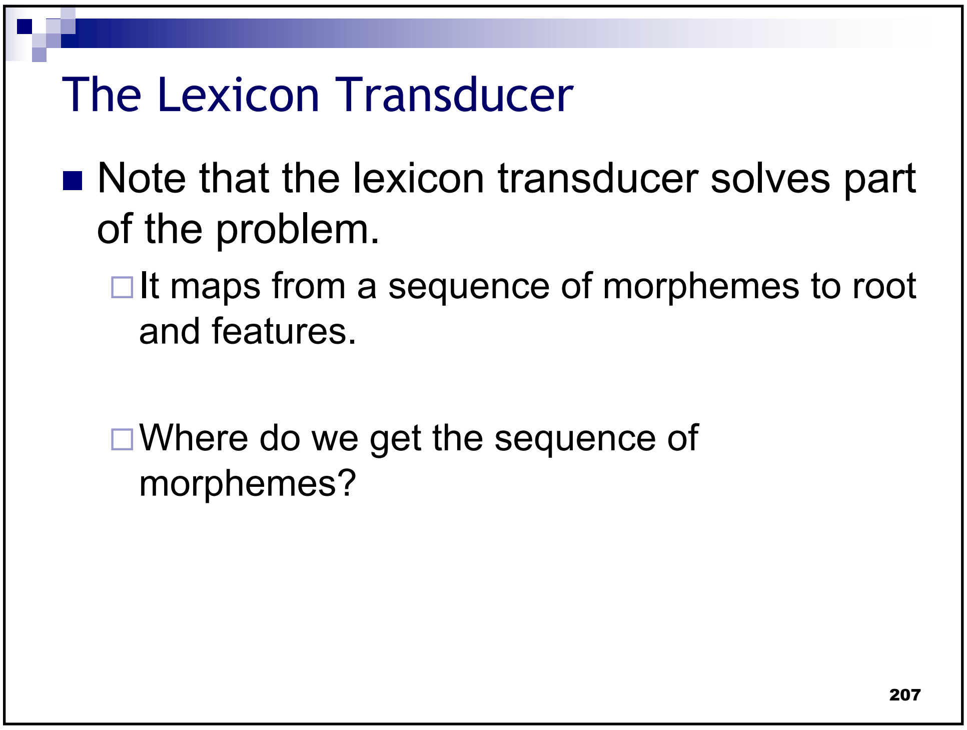
Arabic Morphology on Web

- Play with XRCE's Arabic Analyzer at
  - □ https://open.xerox.com/Services/arabic-morphology

Lexicon as a FS Transducer

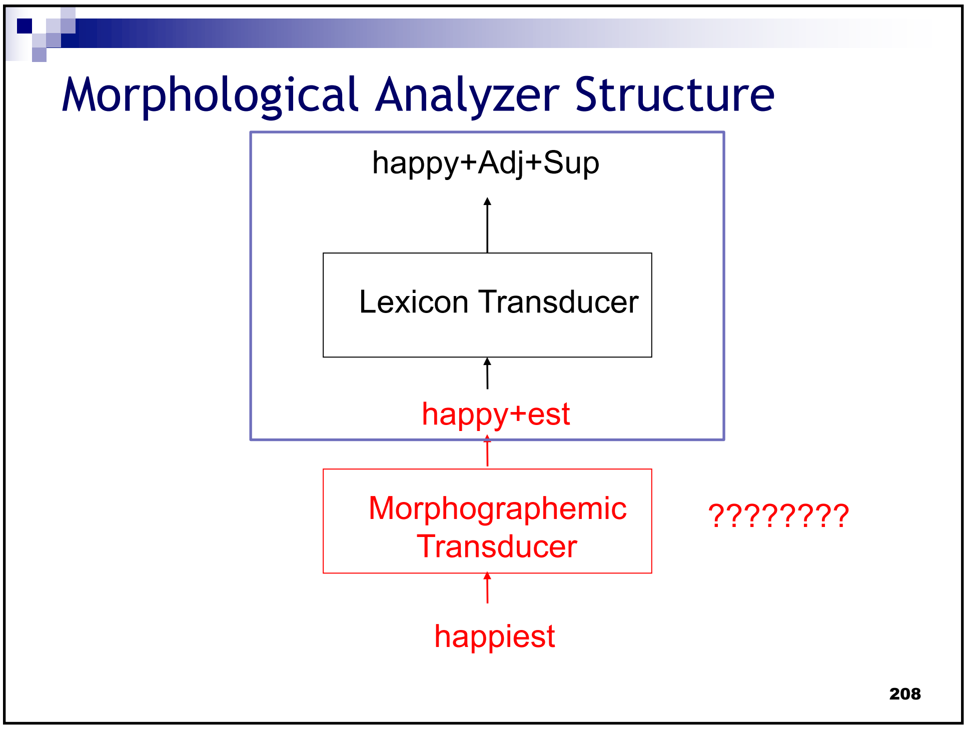


Nondeterminism



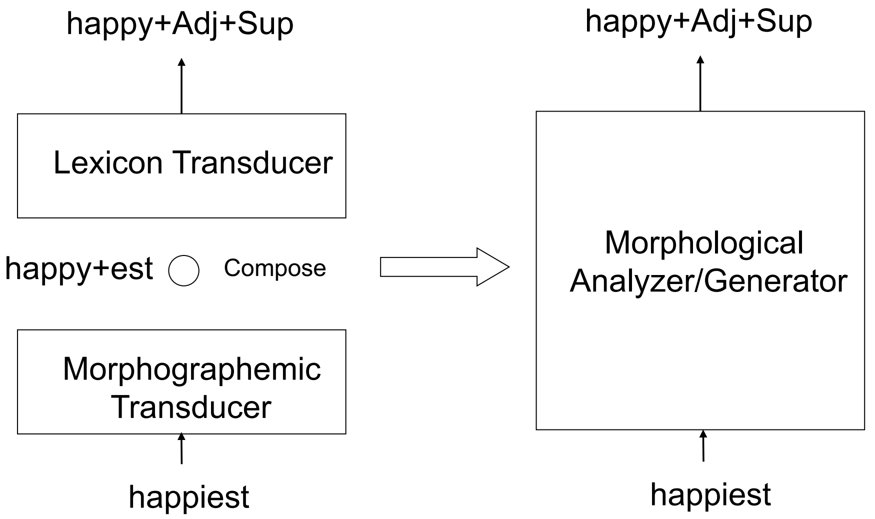
The Lexicon Transducer

- Note that the lexicon transducer solves part of the problem.
  - □ It maps from a sequence of morphemes to root and features.
  - □ Where do we get the sequence of morphemes?



Morphological Analyzer Structure

Sneak Preview (of things to come)



happy+Adj+Sup

Lexicon Transducer

happy+est ○ Compose

Morphographemic Transducer

happiest

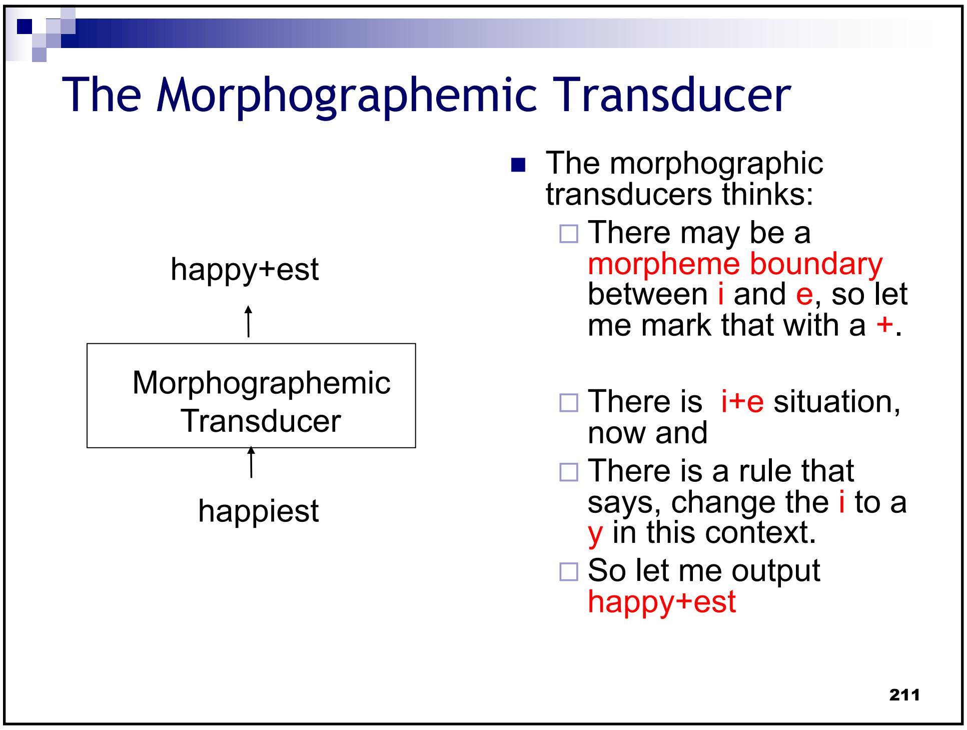
happy+Adj+Sup

Morphological Analyzer/Generator

happiest

The Morphographemic Transducer

- The morphographemic transducer generates
  - all possible ways the input word can be segmented and “unmangled”
  - As sanctioned by the alternation rules of the language
    - Graphemic conventions
    - Morphophonological processes (reflected to the orthography)



The Morphographemic Transducer

happy+est

↑

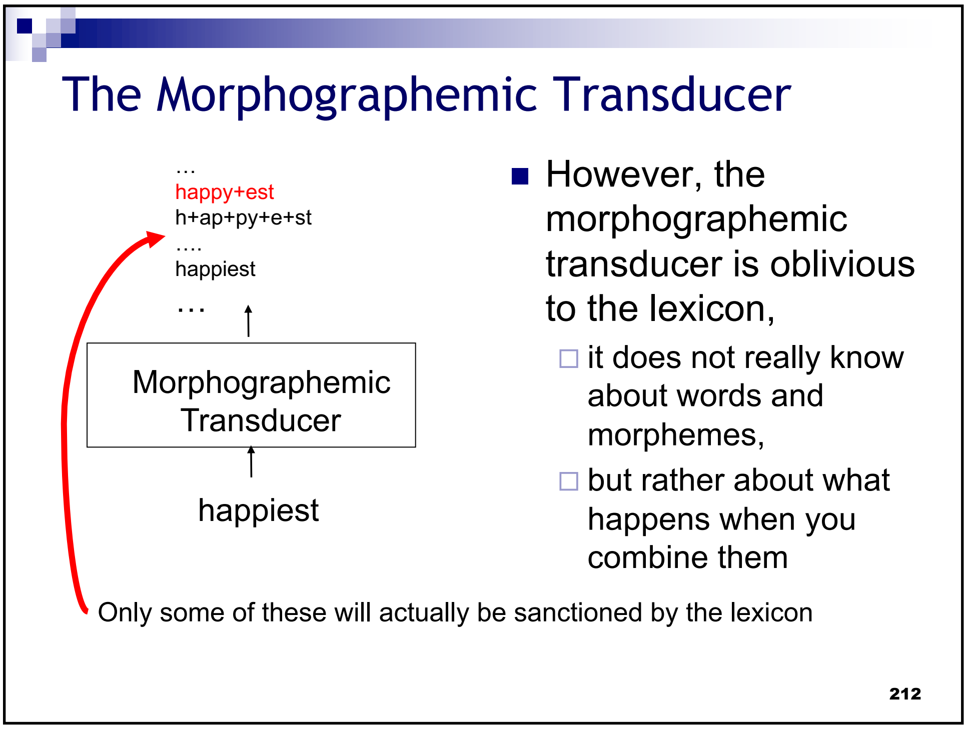
Morphographemic Transducer

↑

happiest

- The morphographic transducers thinks:
  - □ There may be a morpheme boundary between i and e, so let me mark that with a +.
  - □ There is i+e situation, now and
  - □ There is a rule that says, change the i to a y in this context.
  - □ So let me output happy+est

211



The Morphographemic Transducer

…
happy+est
h+ap+py+e+st
….
happiest
…

↑

Morphographemic Transducer

↑

happiest

- However, the morphographemic transducer is oblivious to the lexicon,
  - □ it does not really know about words and morphemes,
  - □ but rather about what happens when you combine them

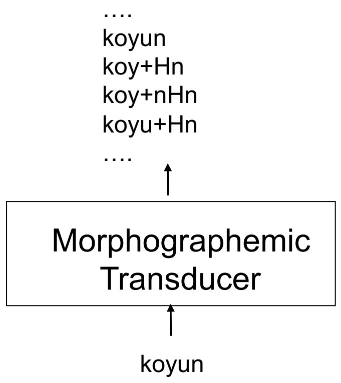
Only some of these will actually be sanctioned by the lexicon

212

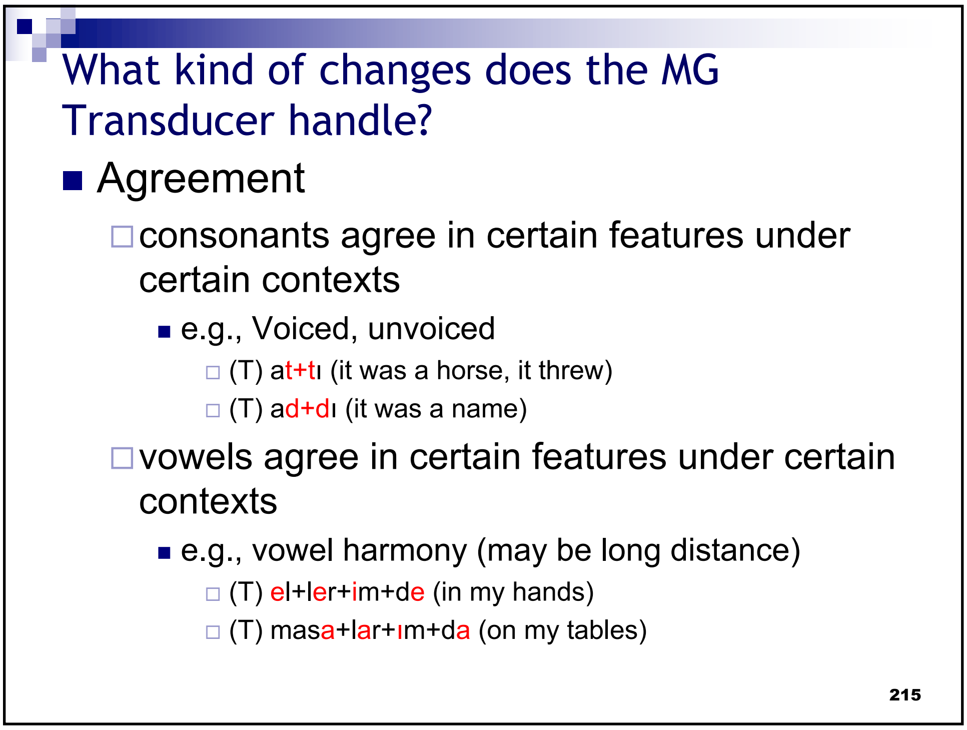
The Morphographemic Transducer

- This obliviousness is actually a good thing
  - □ Languages easily import, generate new words
  - □ But not necessarily such rules! (and usually there are a "small" number of rules.)
  - □ brag⇒bragged, flog ⇒ flogged
    - In a situation like vowel g + vowel insert another g
  - □ And you want these rules to also apply to new words coming to the lexicon
    - blog ⇒ blogged, vlog ⇒vlogged, etc.

The Morphographemic Transducer



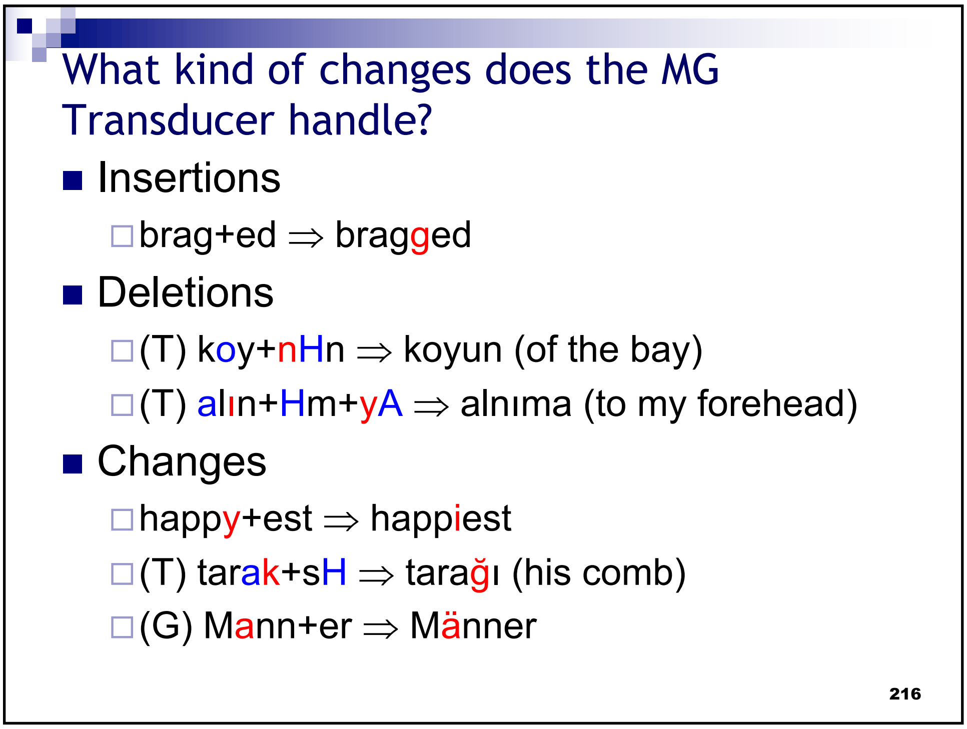
- Also, especially in languages that allow segmentation ambiguity, there may be multiple legitimate outputs of the transducer that are sanctioned by the lexicon transducer.



What kind of changes does the MG Transducer handle?

- Agreement
  - consonants agree in certain features under certain contexts
    - e.g., Voiced, unvoiced
      - (T) at+tı (it was a horse, it threw)
      - (T) ad+dı (it was a name)
  - vowels agree in certain features under certain contexts
    - e.g., vowel harmony (may be long distance)
      - (T) el+ler+im+de (in my hands)
      - (T) masa+lar+ım+da (on my tables)

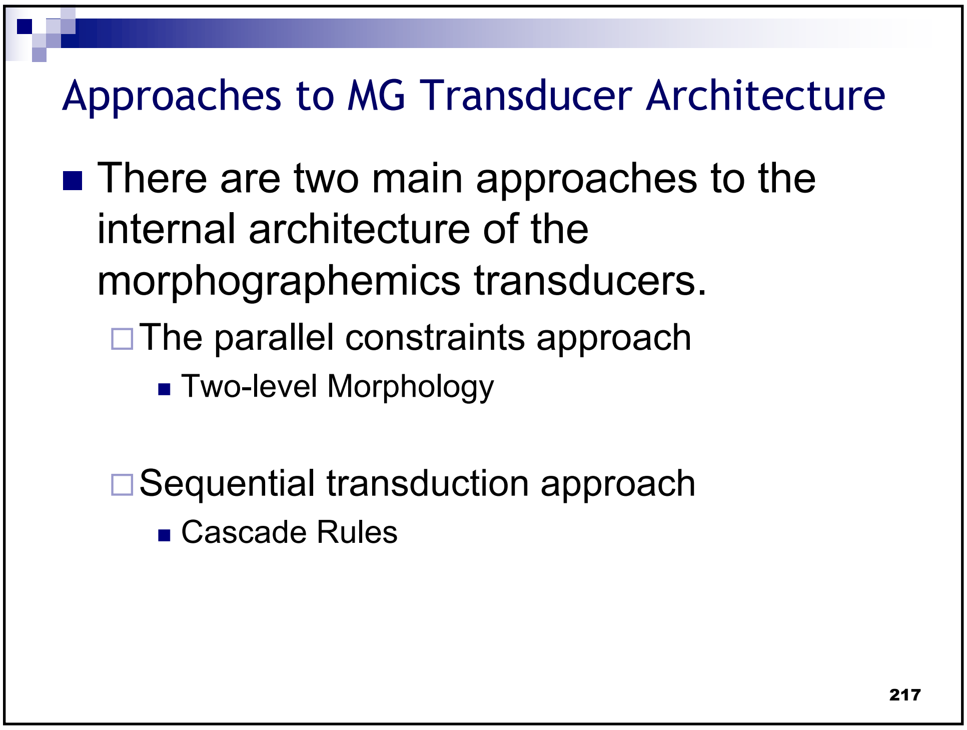
215



What kind of changes does the MG Transducer handle?

- Insertions
  - brag+ed ⇒ bragged
- Deletions
  - (T) koy+nHn ⇒ koyun (of the bay)
  - (T) alın+Hm+yA ⇒ alnıma (to my forehead)
- Changes
  - happy+est ⇒ happiest
  - (T) tarak+sH ⇒ tarağı (his comb)
  - (G) Mann+er ⇒ Männer

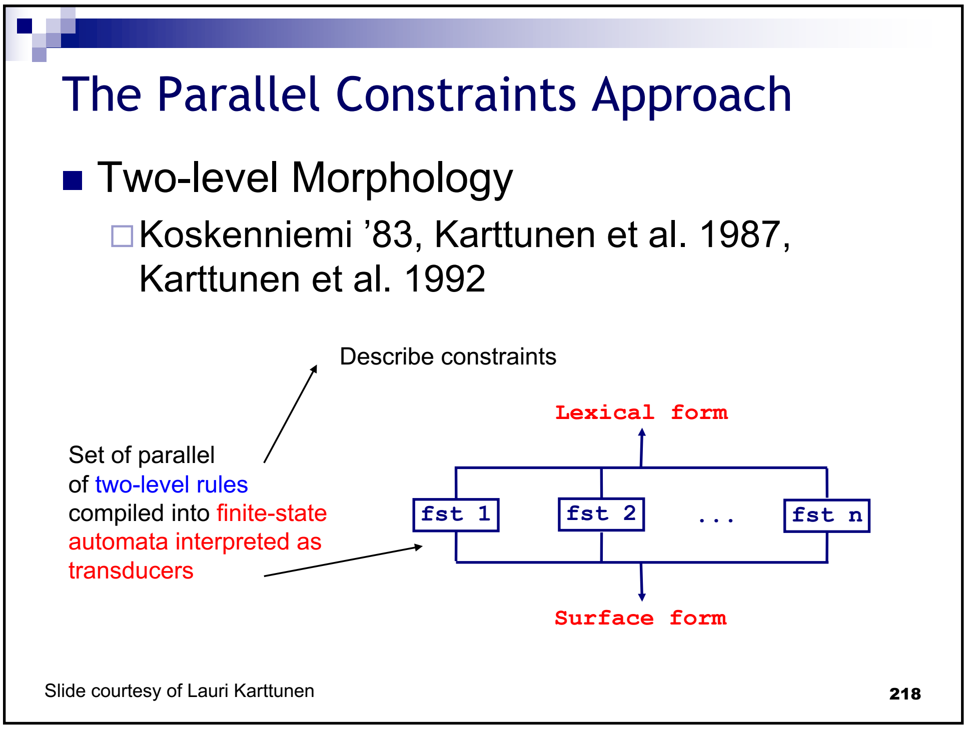
216



Approaches to MG Transducer Architecture

- There are two main approaches to the internal architecture of the morphographemics transducers.
  - □ The parallel constraints approach
    - Two-level Morphology
  - □ Sequential transduction approach
    - Cascade Rules

217



The Parallel Constraints Approach

- Two-level Morphology
  - □ Koskenniemi '83, Karttunen et al. 1987, Karttunen et al. 1992

Describe constraints

Set of parallel of two-level rules compiled into finite-state automata interpreted as transducers

Lexical form

fst 1

fst 2

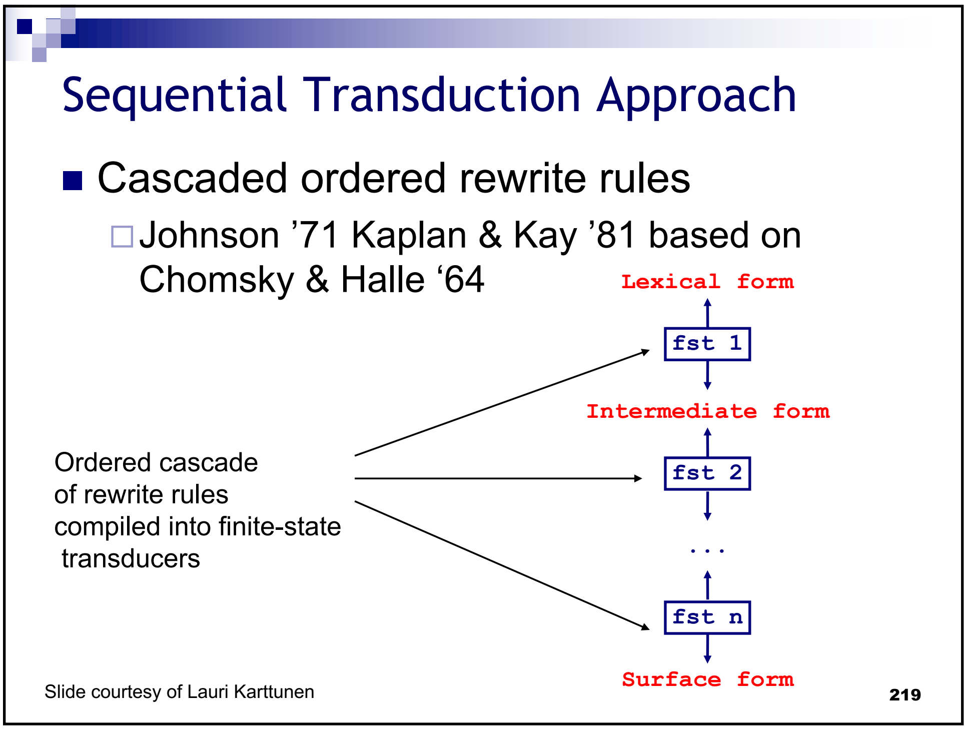
. . .

fst n

Surface form

Slide courtesy of Lauri Karttunen

218



Sequential Transduction Approach

- Cascaded ordered rewrite rules
  - Johnson ’71 Kaplan & Kay ’81 based on Chomsky & Halle ‘64

Ordered cascade of rewrite rules compiled into finite-state transducers

Lexical form

fst 1

Intermediate form

fst 2

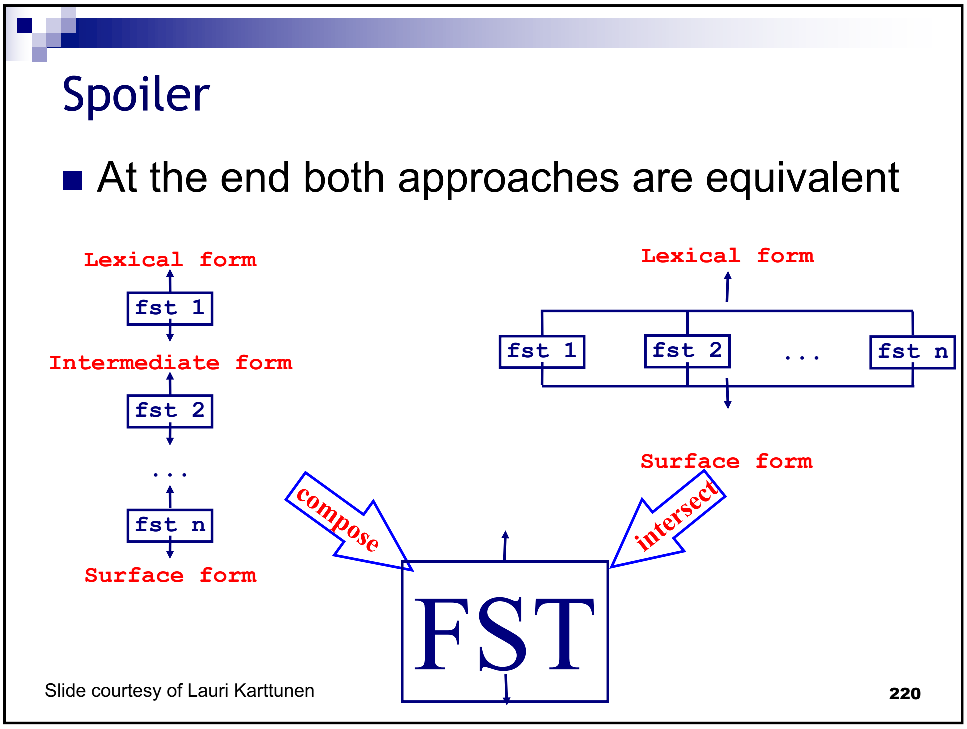
. . .

fst n

Surface form

Slide courtesy of Lauri Karttunen

219



Spoiler

- At the end both approaches are equivalent

Lexical form

fst 1

Intermediate form

fst 2

. . .

fst n

Surface form

compose

Lexical form

fst 1

fst 2

. . .

fst n

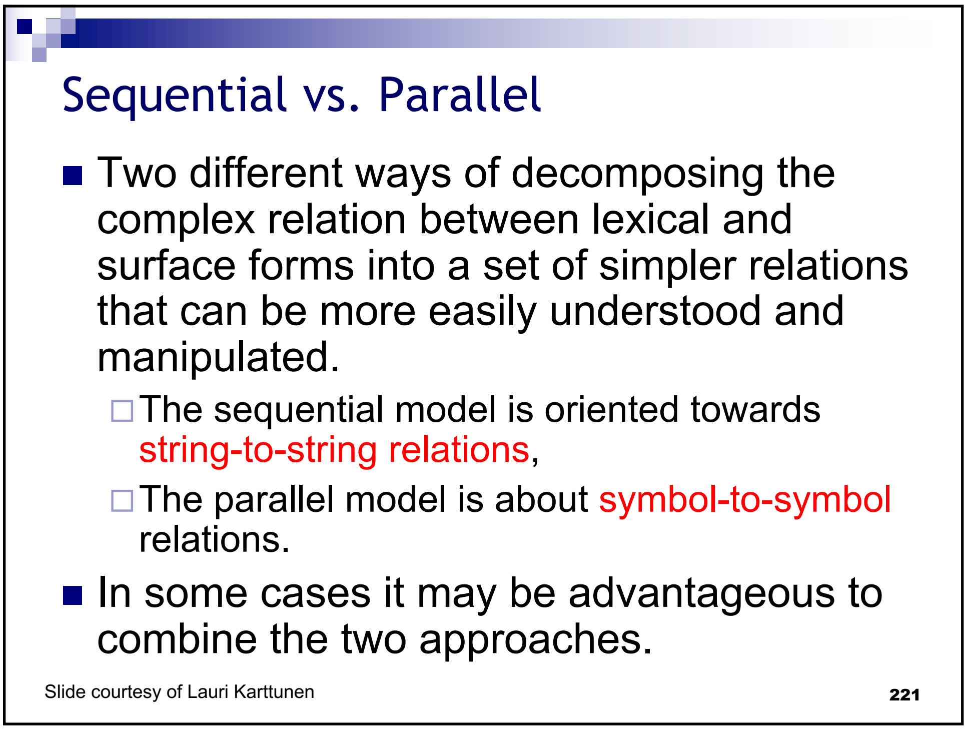
Surface form

intersect

FST

Slide courtesy of Lauri Karttunen

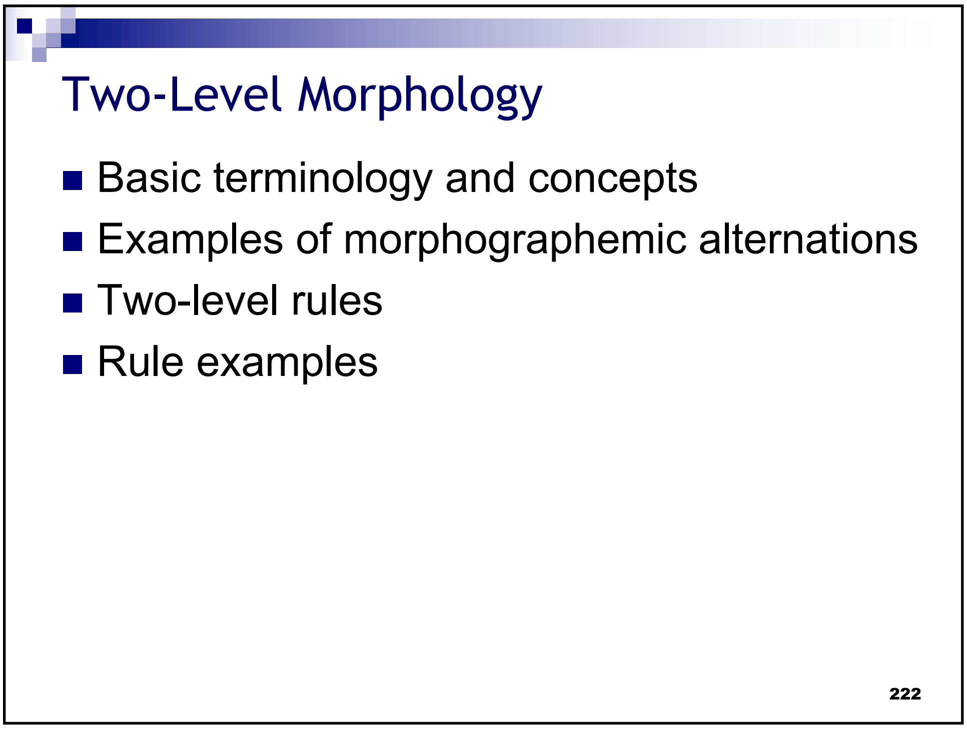
220



Sequential vs. Parallel

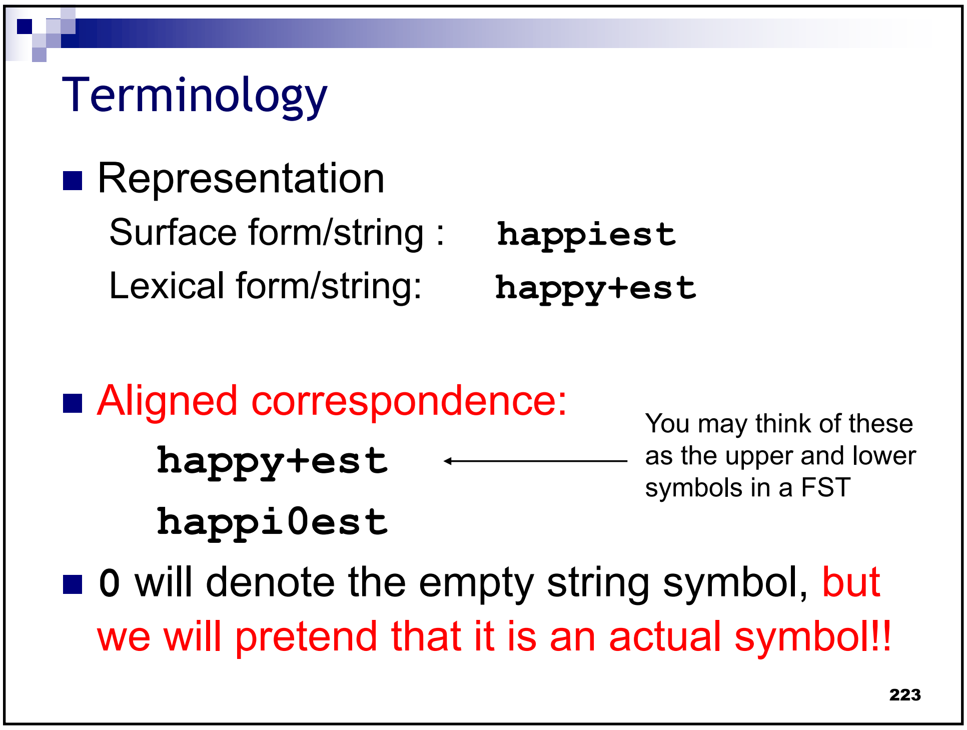
- Two different ways of decomposing the complex relation between lexical and surface forms into a set of simpler relations that can be more easily understood and manipulated.
  - □ The sequential model is oriented towards string-to-string relations,
  - □ The parallel model is about symbol-to-symbol relations.
- In some cases it may be advantageous to combine the two approaches.

Slide courtesy of Lauri Karttunen



Two-Level Morphology

- Basic terminology and concepts
- Examples of morphographemic alternations
- Two-level rules
- Rule examples



Terminology

- Representation

  Surface form/string : **`happiest`**

  Lexical form/string: **`happy+est`**

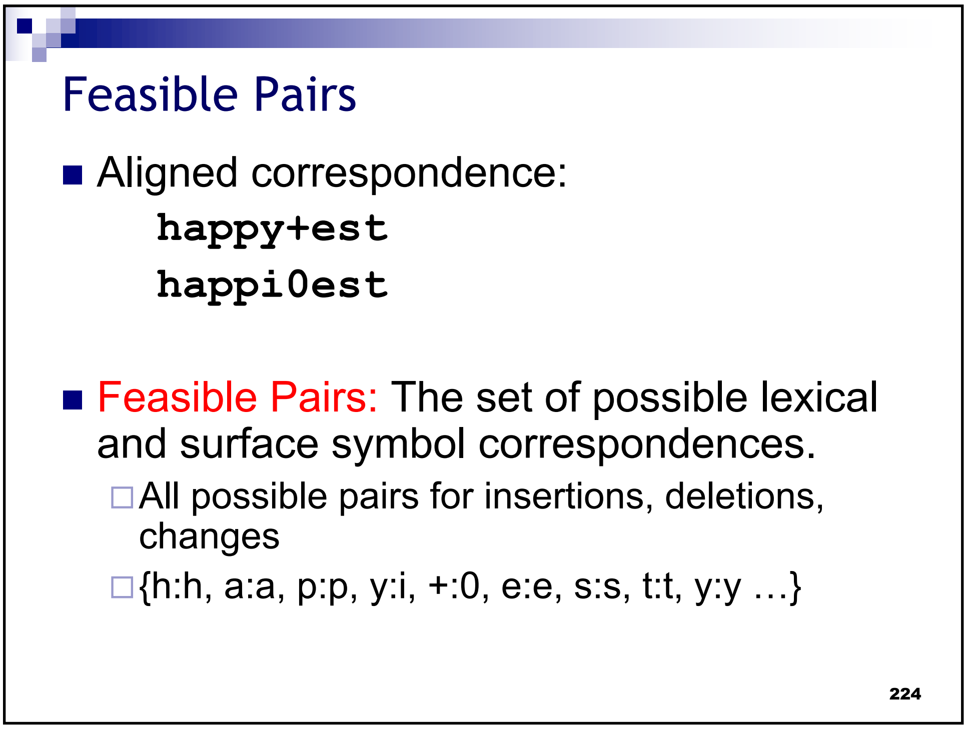
- Aligned correspondence:

  `happy+est`

  `happi0est`

  You may think of these as the upper and lower symbols in a FST

- `0` will denote the empty string symbol, but we will pretend that it is an actual symbol!!



Feasible Pairs

- Aligned correspondence:

  `happy+est`

  `happi0est`

- Feasible Pairs: The set of possible lexical and surface symbol correspondences.
  - □ All possible pairs for insertions, deletions, changes
  - □ {h:h, a:a, p:p, y:i, +:0, e:e, s:s, t:t, y:y …}

Aligned Correspondence

- Aligned correspondence:

h	a	p	p	y	+	e	s	t
h	a	p	p	i	0	e	s	t

- The alignments can be seen as
  - □ Strings in a regular language over the alphabet of feasible pairs, (i.e., symbols that look like “y:i”) or

Aligned Correspondence

- Aligned correspondence:

```
happy+est
happi0est
```

- The alignments can be seen as
  - □ Strings in a regular language over the alphabet of feasible pairs, (i.e., symbols that look like “y:i”) or
  - □ Transductions from surface strings to lexical strings (analysis), or

Aligned Correspondence

- Aligned correspondence:

  `happy+est`

  `happi0est`

- The alignments can be seen as
  - □ Strings in a regular language over the alphabet of feasible pairs, (i.e., symbols that look like "y:i") or
  - □ Transductions from surface strings to lexical strings (analysis), or
  - □ Transductions from lexical strings to surface strings (generation)

227

The Morphographemic Component

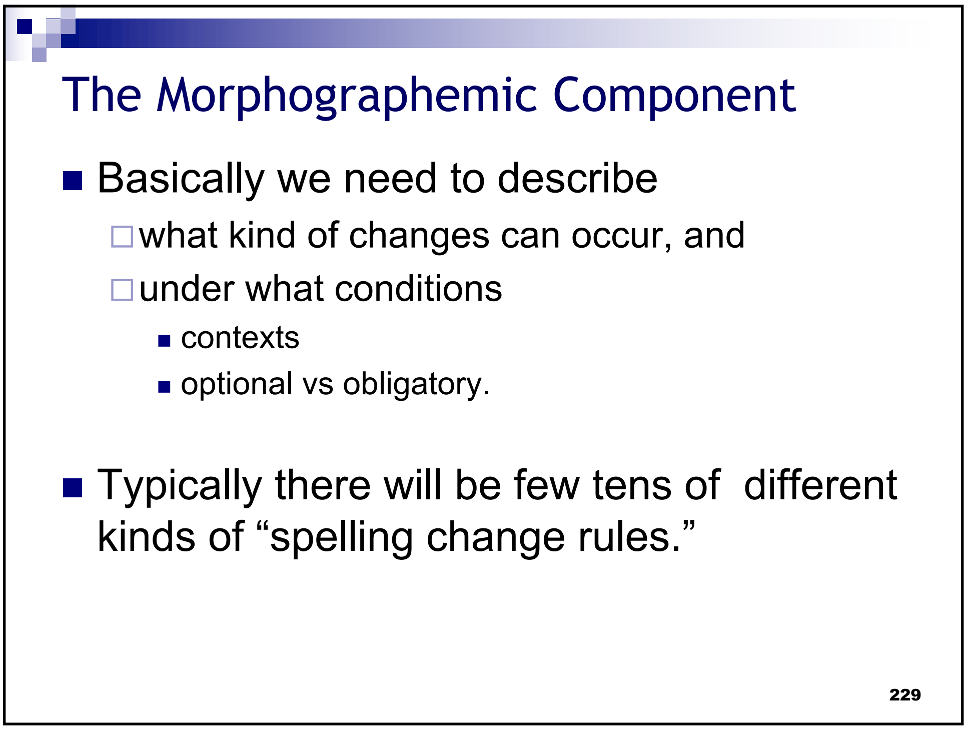
- So how do we define this regular language (over the alphabet of feasible pairs) or the transductions?
  - □ We want to accept the pairs

`stop0+ed`	**`try+ing`**	**`try+ed`**
`stopp0ed`	**`try0ing`**	**`tri0ed`**

  - □ but reject

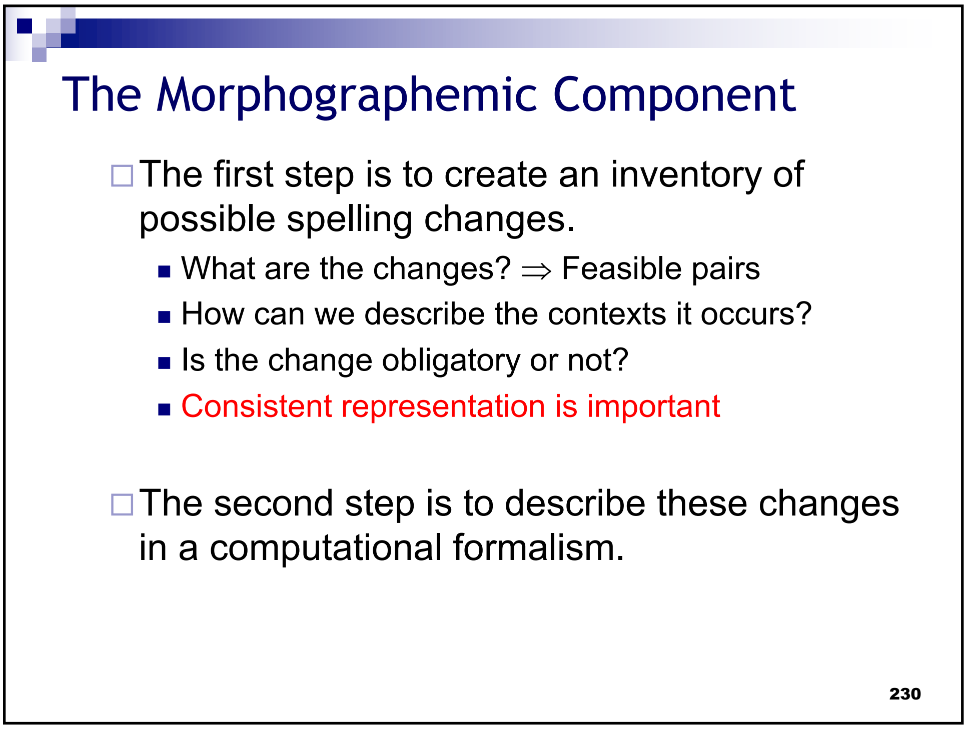
`stop+ed`	**`try+ing`**	**`try+ed`**
`stop0ed`	**`tri0ing`**	**`try0ed`**

228



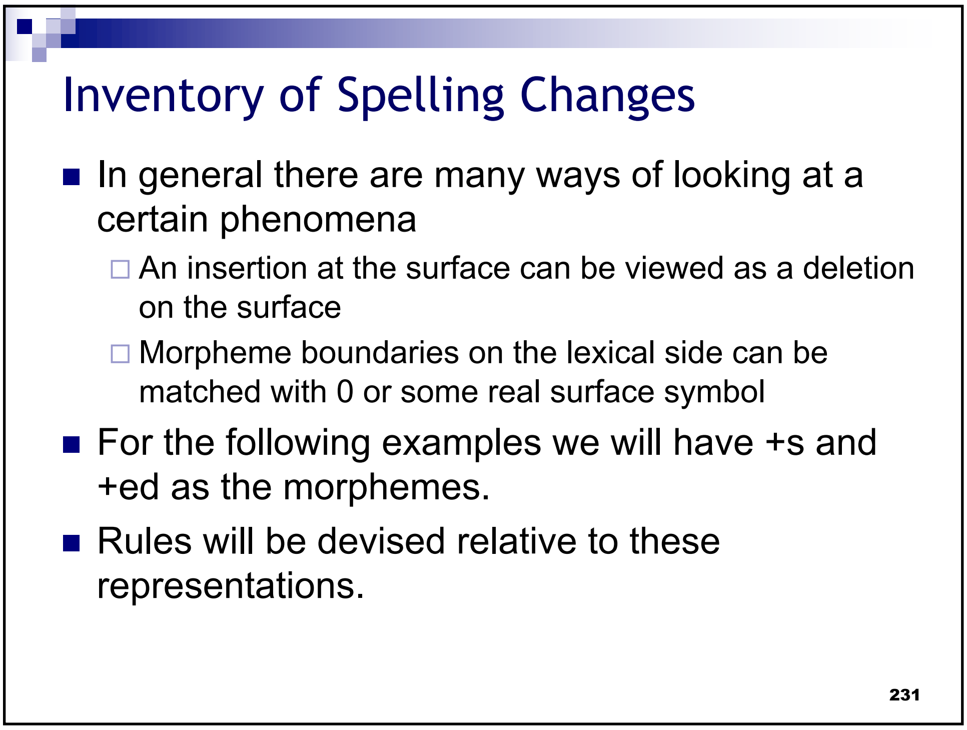
The Morphographemic Component

- Basically we need to describe
  - □ what kind of changes can occur, and
  - □ under what conditions
    - contexts
    - optional vs obligatory.
- Typically there will be few tens of different kinds of “spelling change rules.”



The Morphographemic Component

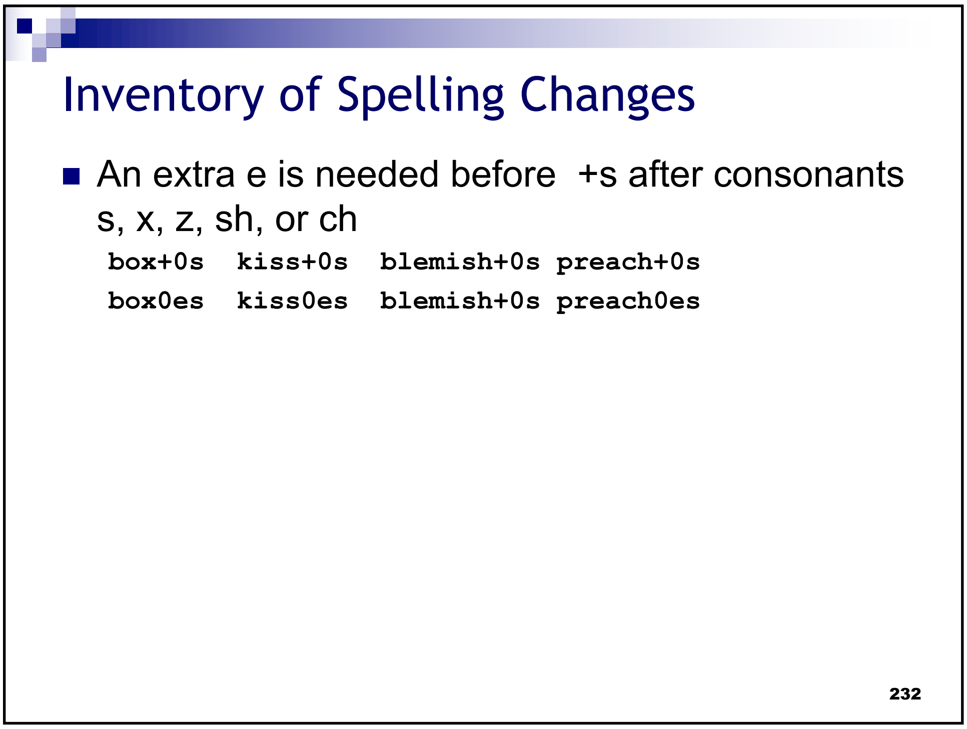
- □ The first step is to create an inventory of possible spelling changes.
  - What are the changes? ⇒ Feasible pairs
  - How can we describe the contexts it occurs?
  - Is the change obligatory or not?
  - Consistent representation is important
- □ The second step is to describe these changes in a computational formalism.



Inventory of Spelling Changes

- In general there are many ways of looking at a certain phenomena
  - □ An insertion at the surface can be viewed as a deletion on the surface
  - □ Morpheme boundaries on the lexical side can be matched with 0 or some real surface symbol
- For the following examples we will have +s and +ed as the morphemes.
- Rules will be devised relative to these representations.

231

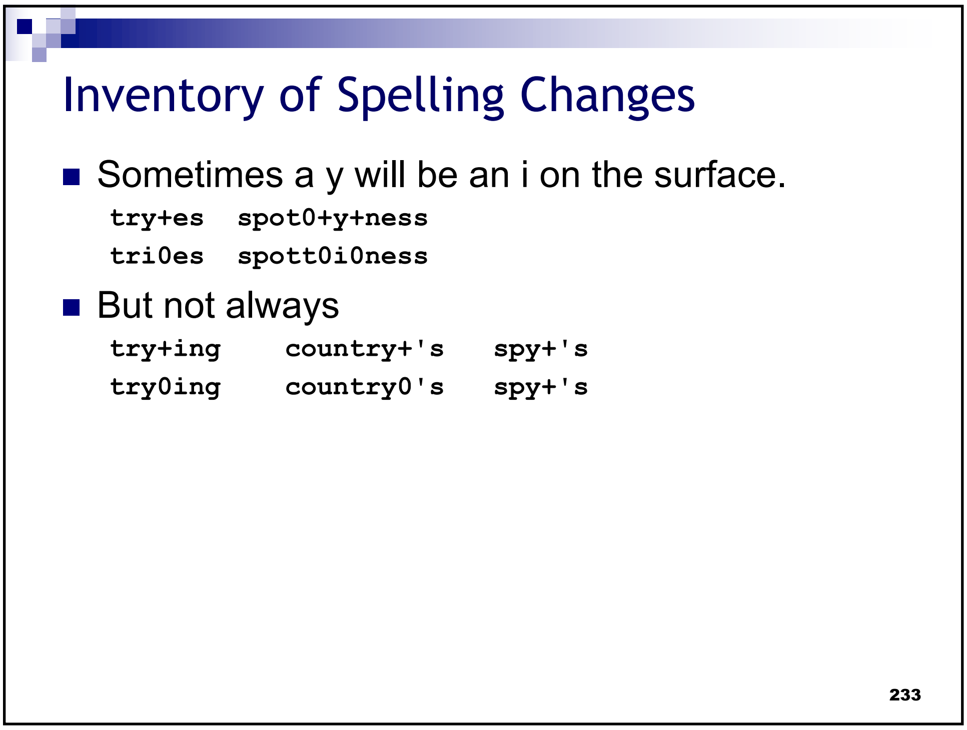


Inventory of Spelling Changes

- An extra e is needed before +s after consonants s, x, z, sh, or ch

```
box+0s   kiss+0s   blemish+0s preach+0s
box0es   kiss0es   blemish+0s preach0es
```

232



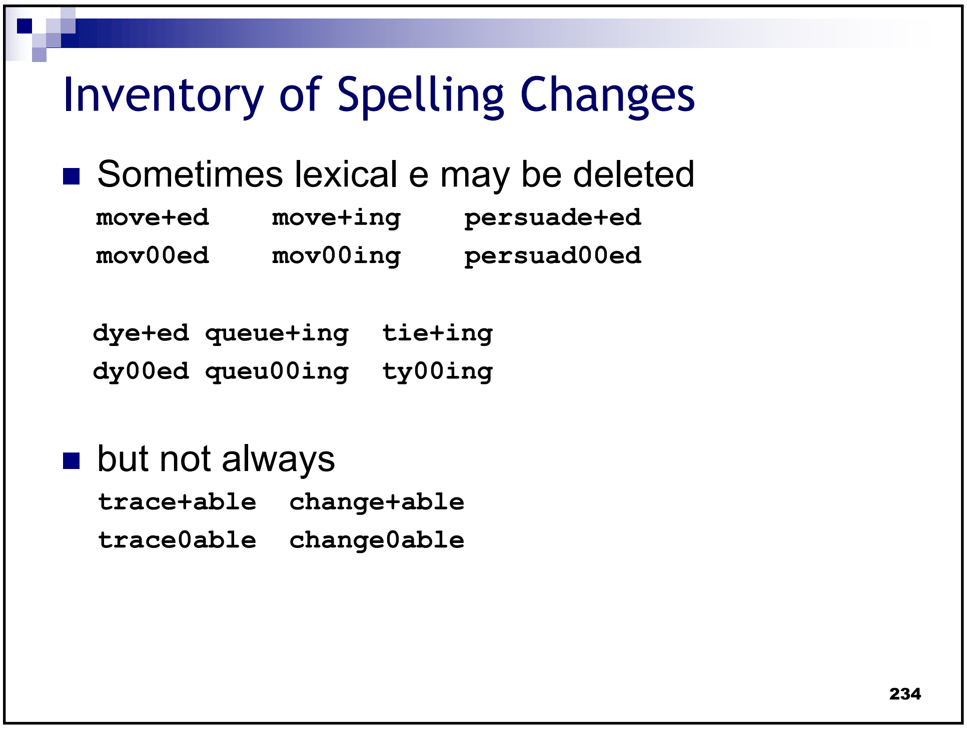
Inventory of Spelling Changes

- Sometimes a y will be an i on the surface.

```
try+es   spot0+y+ness
tri0es   spott0i0ness
```

- But not always

```
try+ing     country+'s    spy+'s
try0ing     country0's    spy+'s
```



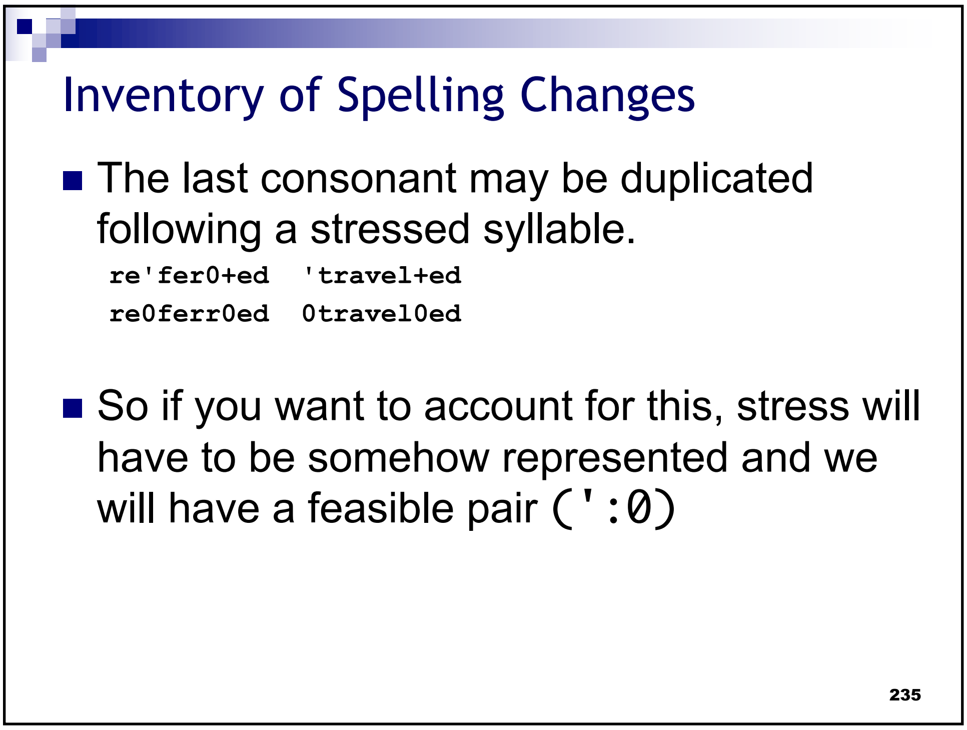
Inventory of Spelling Changes

- Sometimes lexical e may be deleted

```
move+ed     move+ing     persuade+ed
mov00ed     mov00ing     persuad00ed

dye+ed queue+ing  tie+ing
dy00ed queu00ing  ty00ing
```

- but not always

```
trace+able   change+able
trace0able   change0able
```

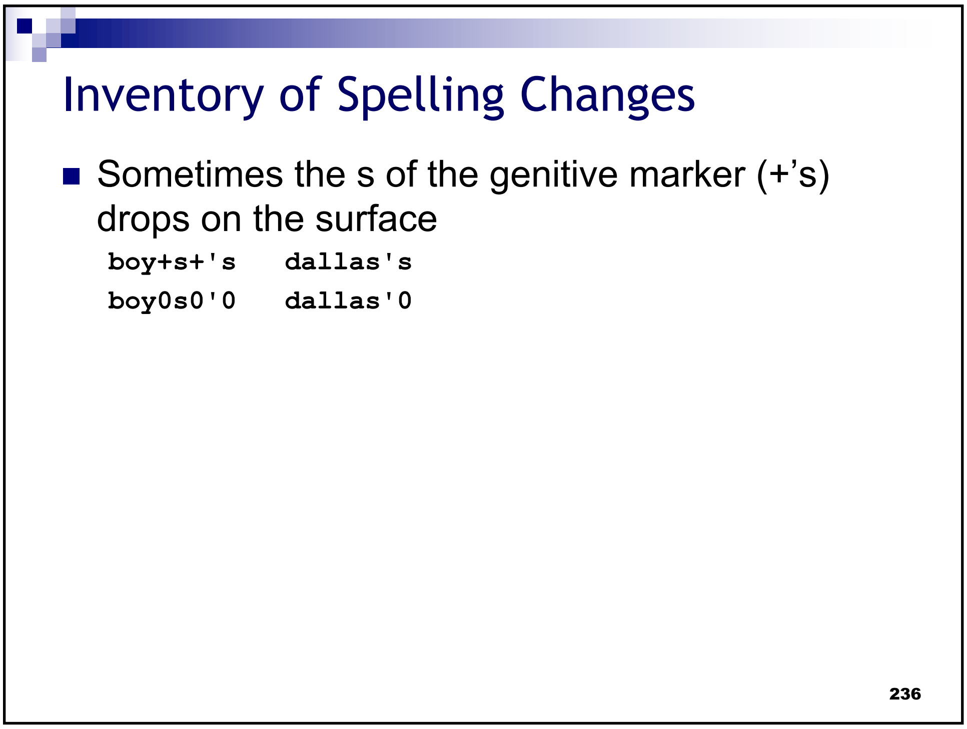


Inventory of Spelling Changes

- The last consonant may be duplicated following a stressed syllable.

```
re'fer0+ed  'travel+ed
re0ferr0ed  0travel0ed
```

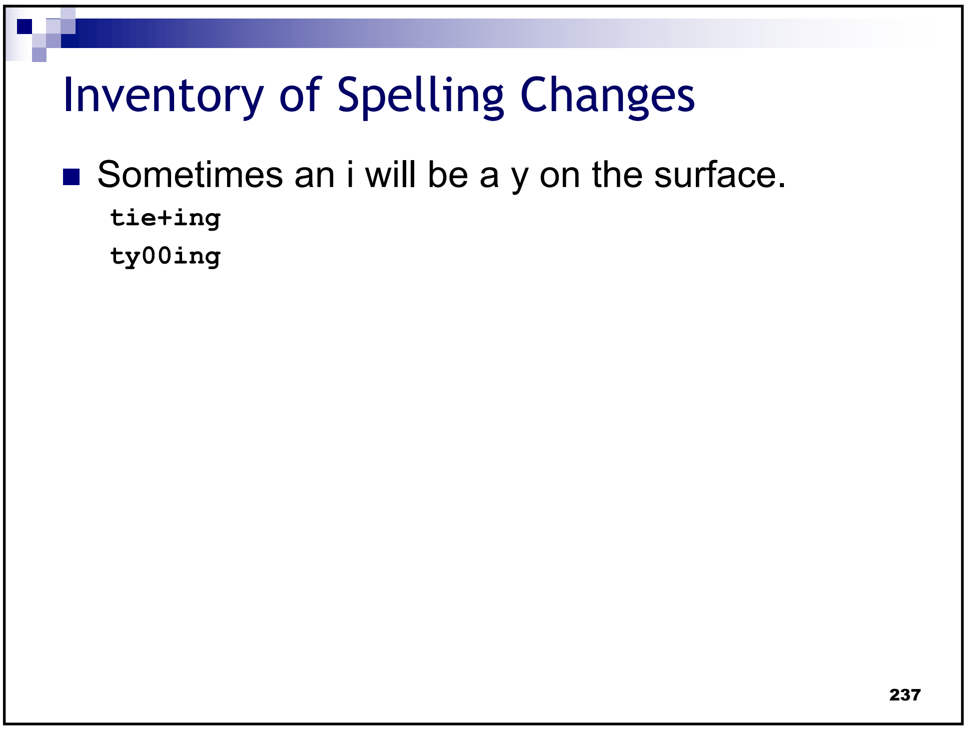
- So if you want to account for this, stress will have to be somehow represented and we will have a feasible pair (':0)



Inventory of Spelling Changes

- Sometimes the s of the genitive marker (+’s) drops on the surface

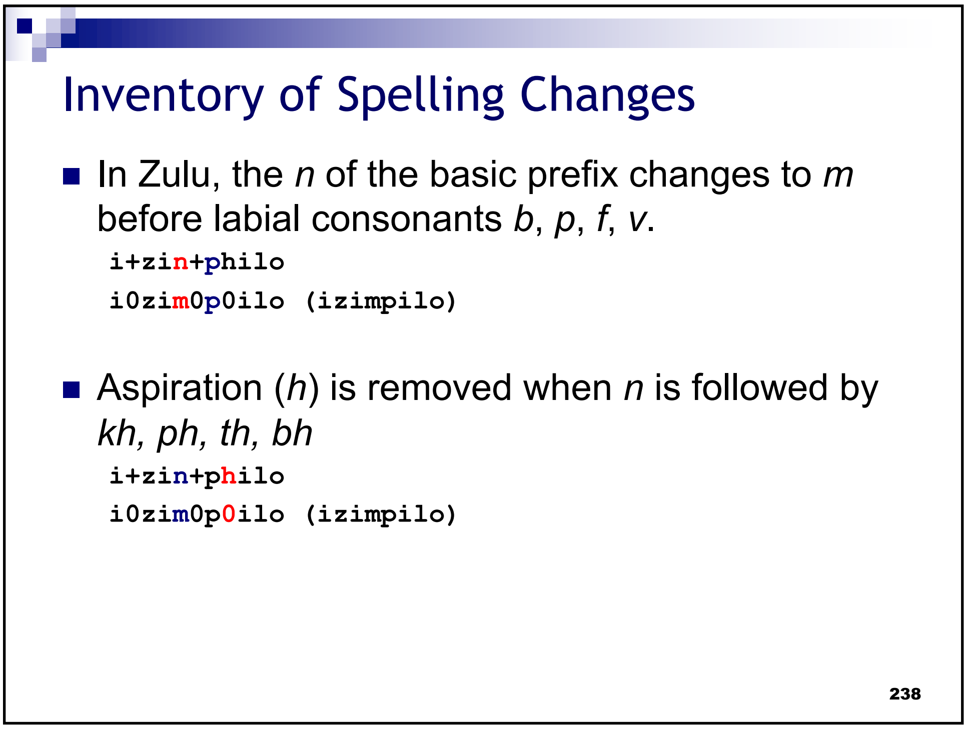
```
boy+s+'s   dallas's
boy0s0'0   dallas'0
```



Inventory of Spelling Changes

- Sometimes an i will be a y on the surface.

```
tie+ing
ty00ing
```



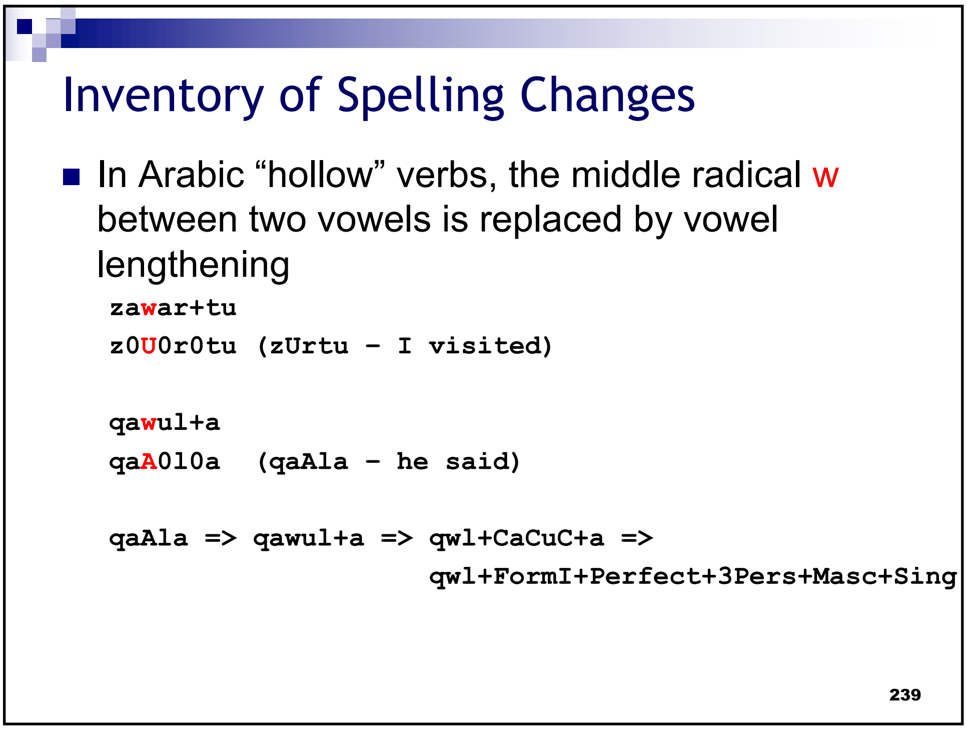
Inventory of Spelling Changes

- In Zulu, the *n* of the basic prefix changes to *m* before labial consonants *b*, *p*, *f*, *v*.

```
i+zin+philo
i0zim0p0ilo (izimpilo)
```

- Aspiration (*h*) is removed when *n* is followed by *kh, ph, th, bh*

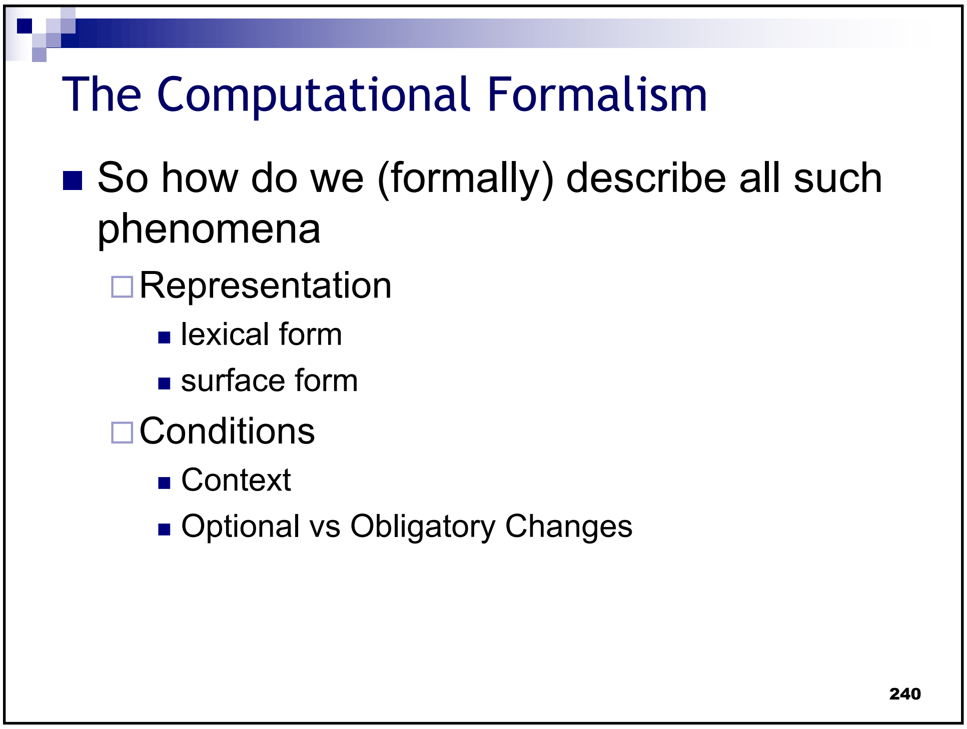
```
i+zin+philo
i0zim0p0ilo (izimpilo)
```



Inventory of Spelling Changes

- In Arabic “hollow” verbs, the middle radical w between two vowels is replaced by vowel lengthening

```
zawar+tu
z0U0r0tu (zUrtu – I visited)

qawul+a
qaA0l0a  (qaAla – he said)

qaAla => qawul+a => qwl+CaCuC+a =>
                    qwl+FormI+Perfect+3Pers+Masc+Sing
```

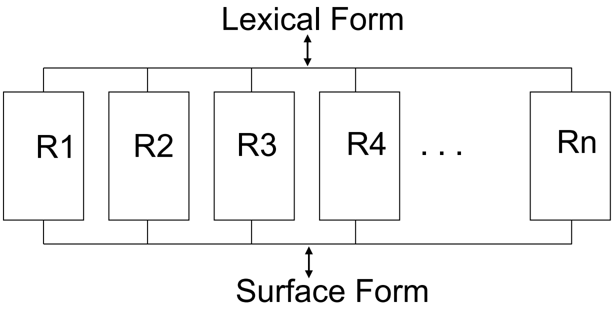


The Computational Formalism

- So how do we (formally) describe all such phenomena
  - □ Representation
    - lexical form
    - surface form
  - □ Conditions
    - Context
    - Optional vs Obligatory Changes

Parallel Rules

- A well-established formalism for describing morphographemic changes.



Lexical Form

R1 R2 R3 R4 . . . Rn

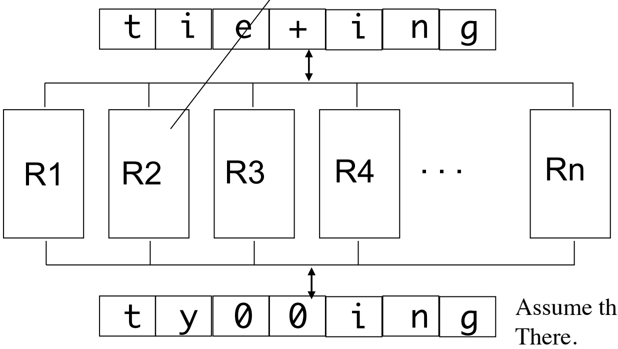
Surface Form

Each rule describes a constraint on legal Lexical - Surface pairings.

All rules operate in parallel!

Parallel Rules

- Each morphographemic constraint is enforced by a finite state recognizer over the alphabet of feasible-pairs.



t i e + i n g

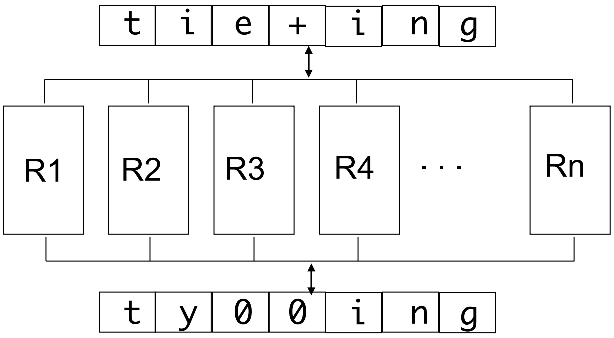
R1 R2 R3 R4 . . . Rn

t y 0 0 i n g

Assume that the 0’s are magically There.
The reason and how are really technical

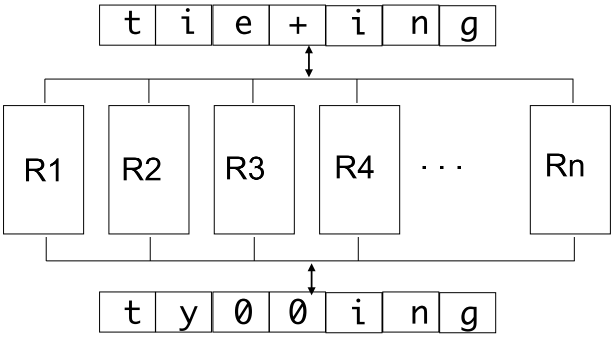
Parallel Rules

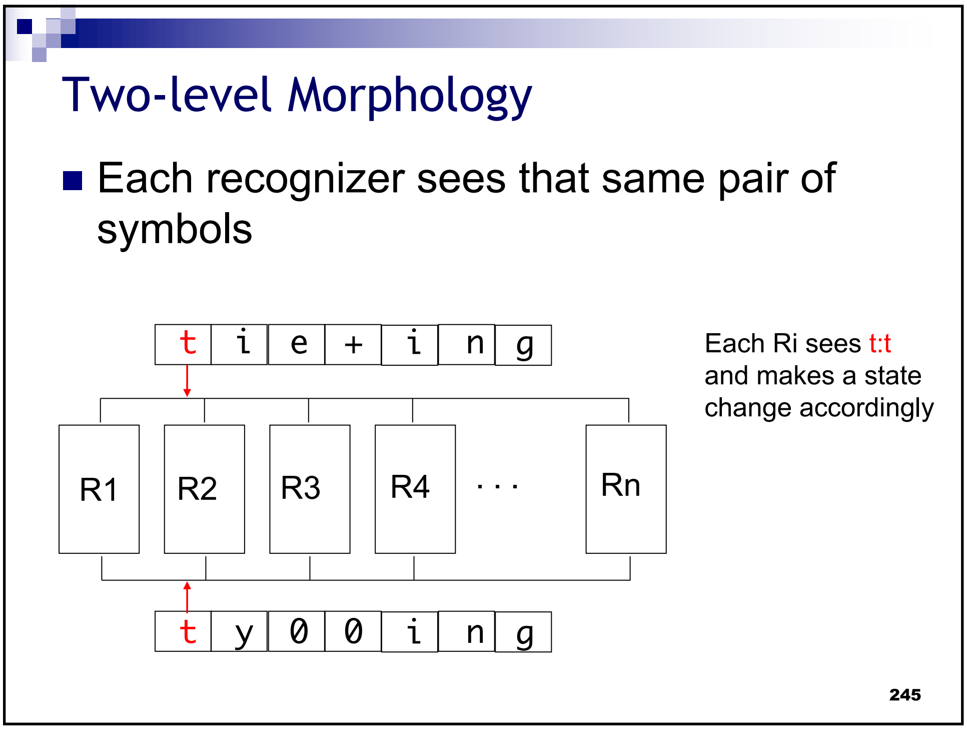
- A lexical-surface string pair is "accepted" if NONE of the rule recognizers reject it.
- Thus, all rules must put a good word in!



Parallel Rules

- Each rule independently checks if it has any problems with the pair of strings.





Two-level Morphology

- Each recognizer sees that same pair of symbols

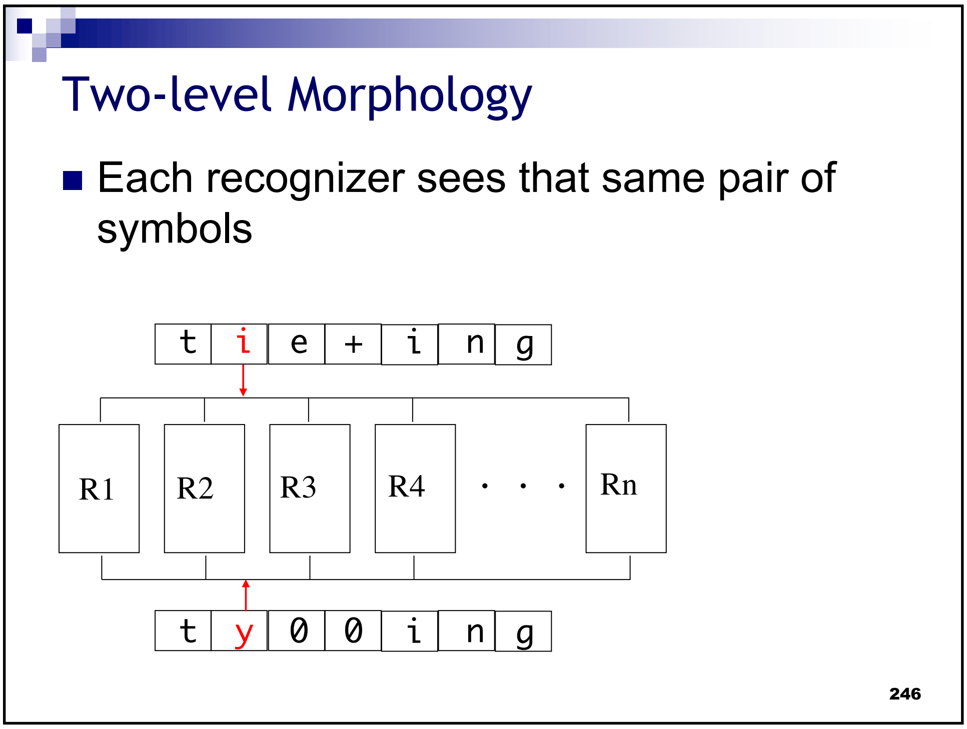
t	i	e	+	i	n	g

R1 R2 R3 R4 · · · Rn

t	y	0	0	i	n	g

Each Ri sees t:t and makes a state change accordingly

245



Two-level Morphology

- Each recognizer sees that same pair of symbols

t	i	e	+	i	n	g

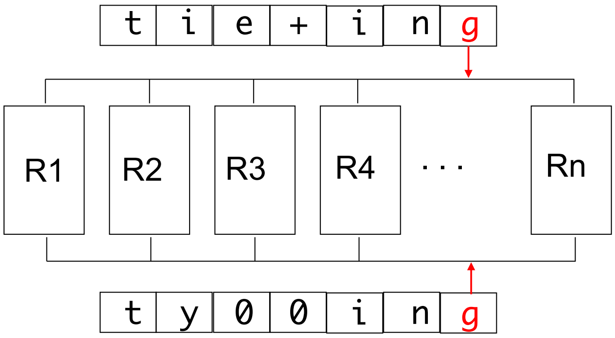
R1 R2 R3 R4 · · · Rn

t	y	0	0	i	n	g

246

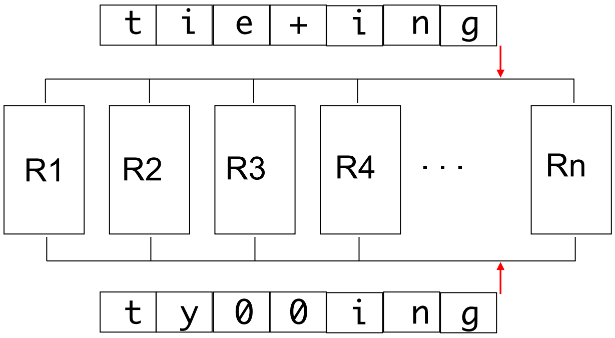
Two-level Morphology

- Each recognizer sees that same pair of symbols



Two-level Morphology

- Each recognizer sees that same pair of symbols

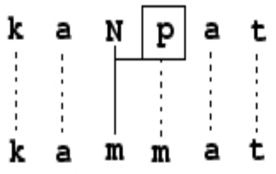


At this point all
Ri should be in
an accepting state.

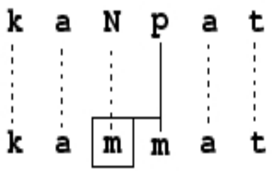
The kaNpat Example

- Language X has a root *kaN*, including an underspecified nasal morphophoneme *N*, that can be followed by the suffix *pat* to produce the well-formed, but still abstract, word *kaNpat*. We may refer to this as the “underlying” or “lexical” or “morphophonemic” form.
- The morphophonology of this language has an alternation: the morphophoneme *N* becomes *m* when it is followed by a *p*.
- In addition, a *p* becomes an *m* when it is preceded by an underlying or derived *m*.
- The “surface” form or “realization” of this word should therefore be *kammat*.

The kaNpat Example

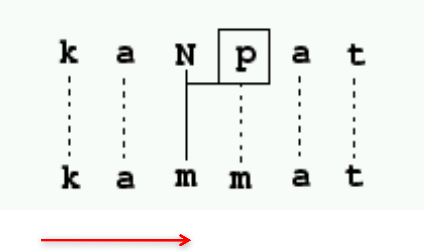


The morphophonology of this language has an alternation: the morphophoneme *N* becomes *m* when it is followed by a *p*.
In addition,



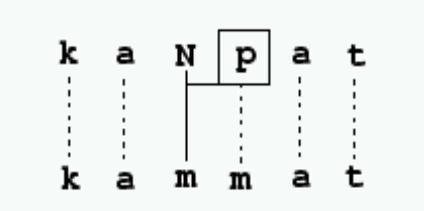
a *p* becomes an *m* when it is preceded by an underlying or derived *m*

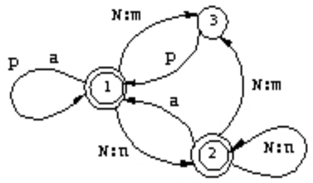
The kaNpat Example



- Ignore everything until you see a N:m pair, and make sure it is followed with some p:@ (some feasible pair with p on the lexical side)
- If you see a N:x (x≠m) then make sure the next pair is not p:@.

The kaNpat Example: N:m rule



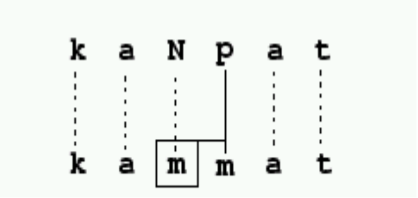


p ⇒ {p:p, p:m} a denotes eveything else

- Ignore everything until you see a N:m pair, and make sure it is followed with some p:@ (some feasible pair with p on the lexical side)
- If you see a N:x (x≠m) then make sure the next pair is not p:@.

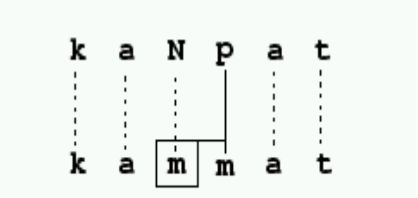
Also remember FST rejects if no arc is found

The kaNpat Example: p:m rule

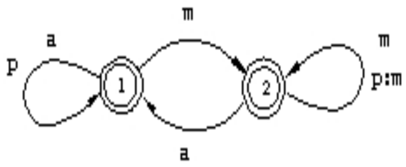


- Make sure that a p:m pairing is preceded by a feasible pair like @:m
- And, no p:x (x ≠ m) follows a @:m

The kaNpat Example: p:m rule

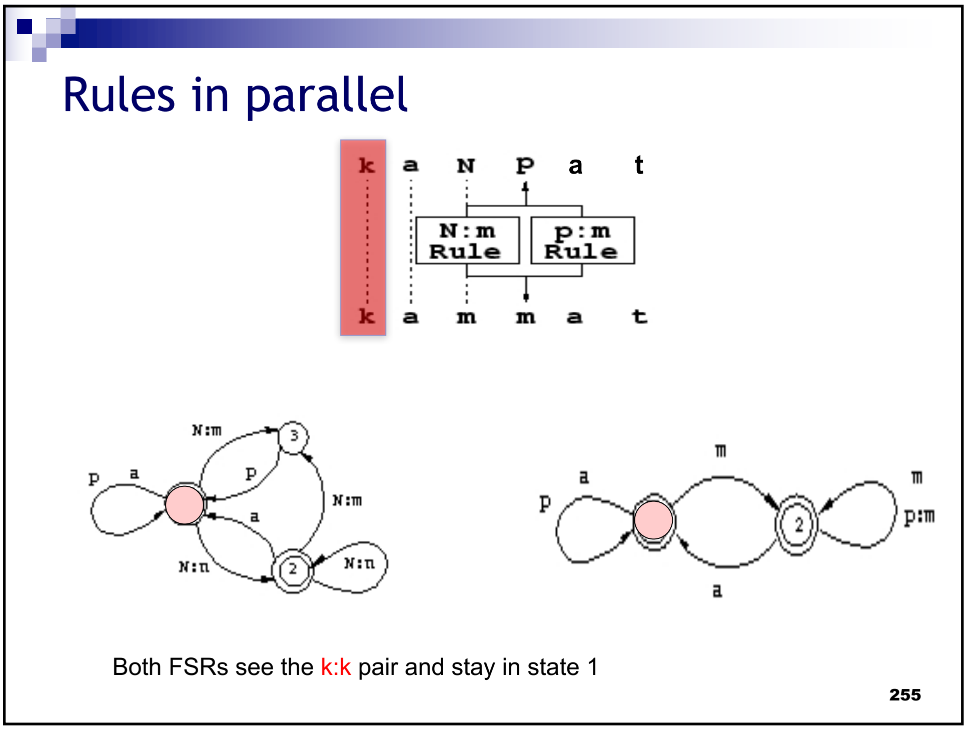


- Make sure that a p:m pairing is preceded by a feasible pair like @:m
- And, no p:x (x ≠ m) follows a @:m



Also remember that if an unlisted input İs encountered, the string is rejected

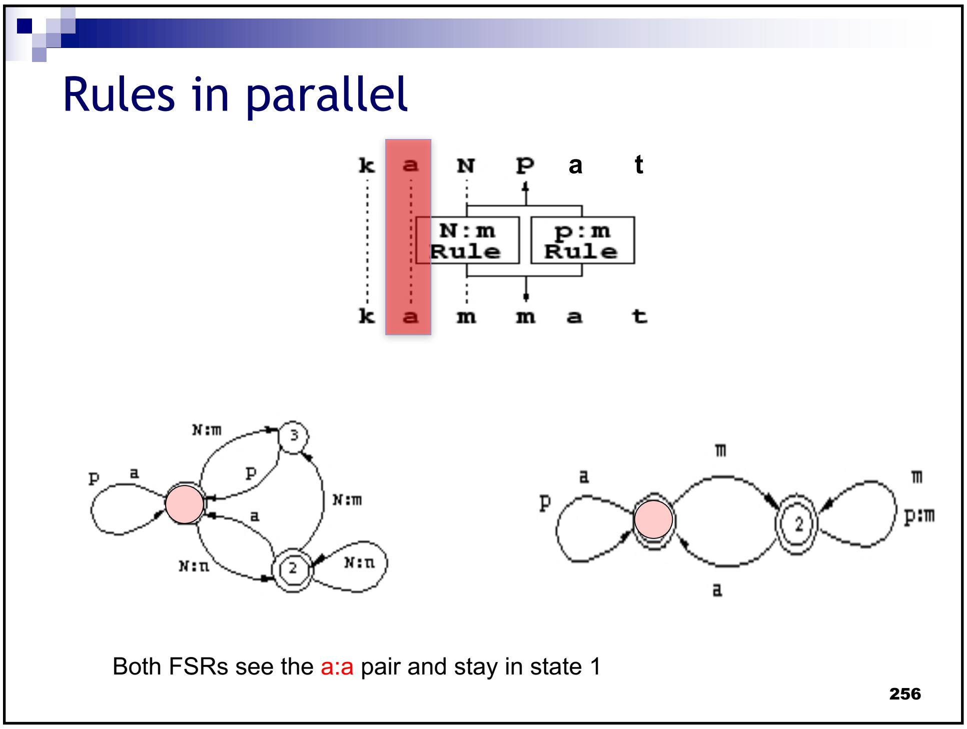
M ⇒{m:m, N:m}, **a everything else**



Rules in parallel

Both FSRs see the k:k pair and stay in state 1

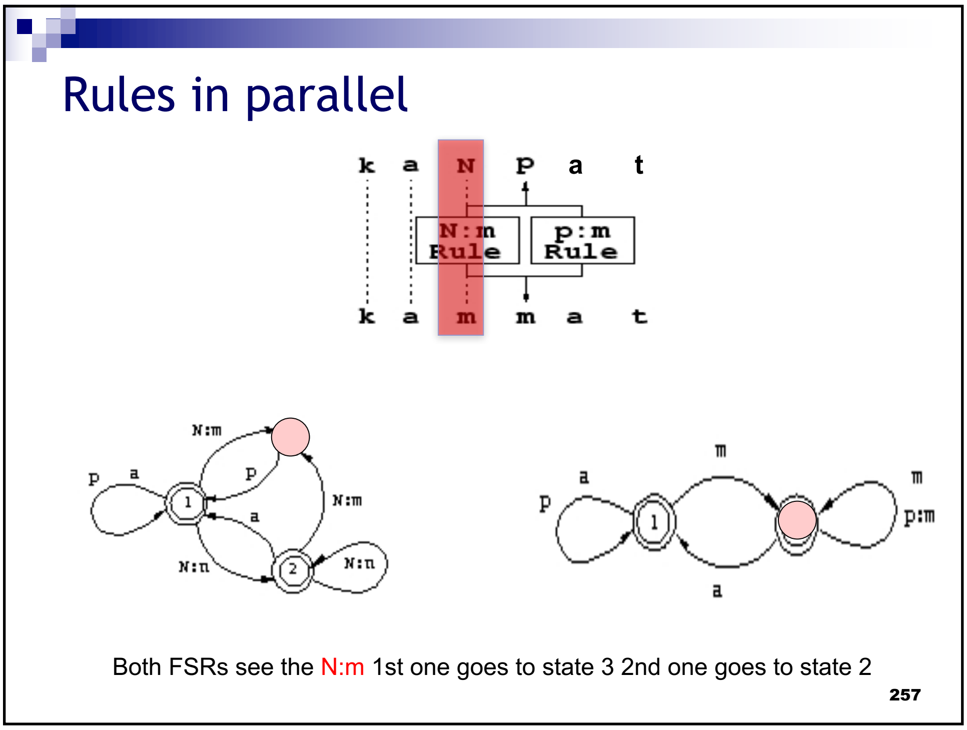
255



Rules in parallel

Both FSRs see the a:a pair and stay in state 1

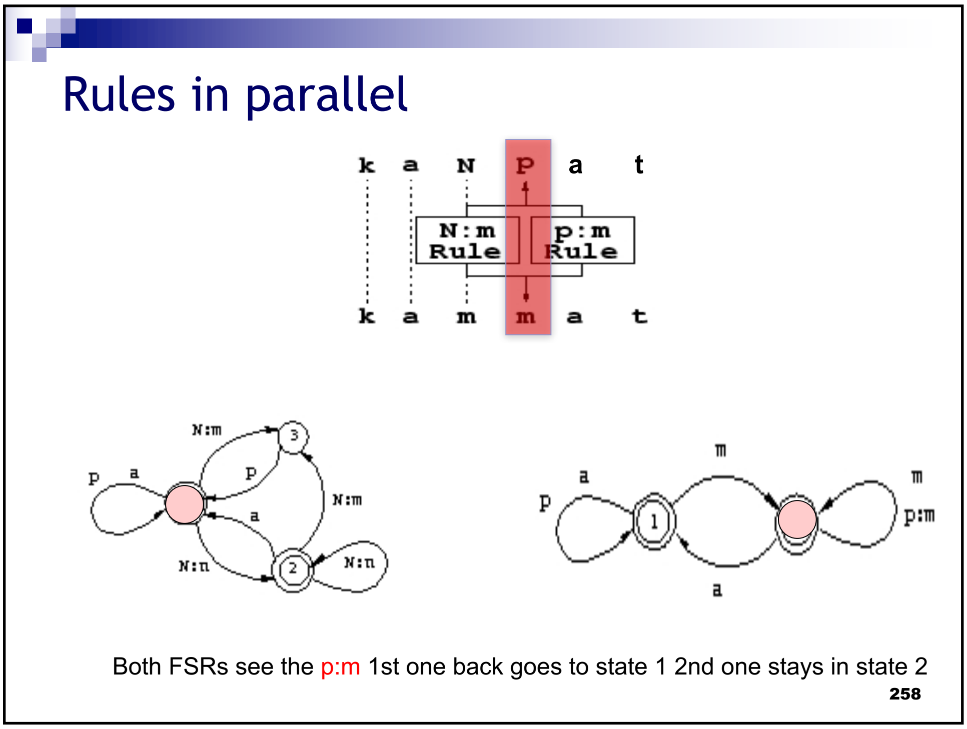
256



Rules in parallel

Both FSRs see the N:m 1st one goes to state 3 2nd one goes to state 2

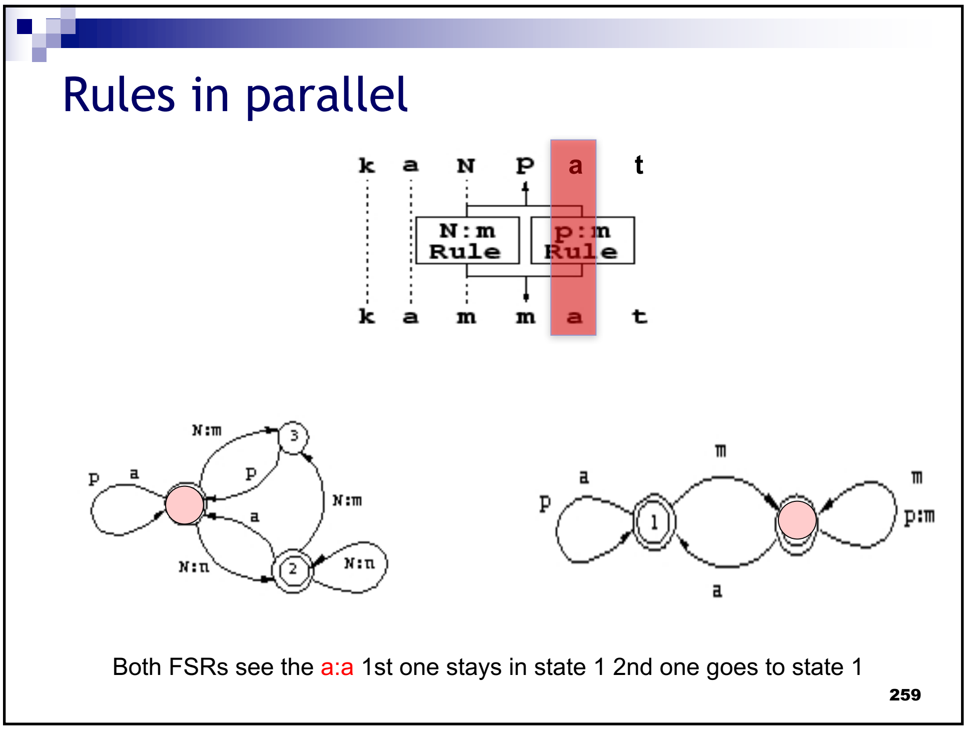
257



Rules in parallel

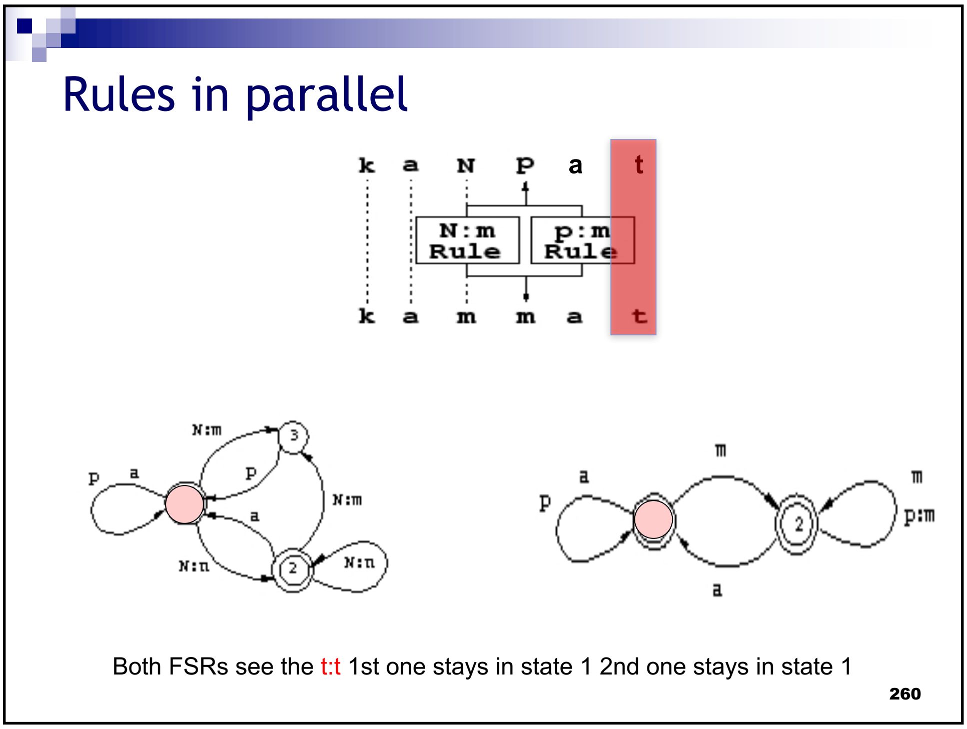
Both FSRs see the p:m 1st one back goes to state 1 2nd one stays in state 2

258



Rules in parallel

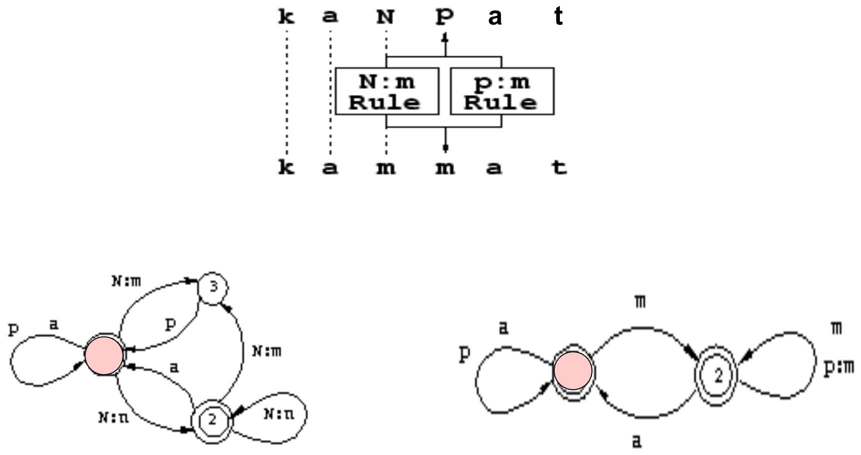
Both FSRs see the a:a 1st one stays in state 1 2nd one goes to state 1



Rules in parallel

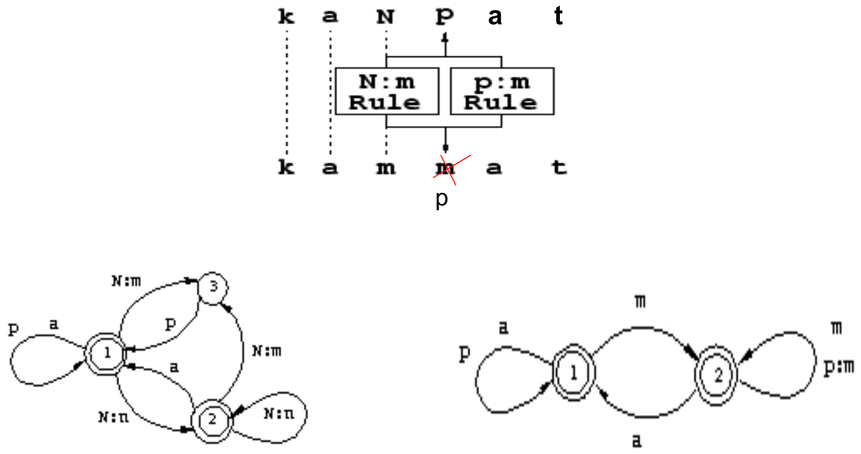
Both FSRs see the t:t 1st one stays in state 1 2nd one stays in state 1

Rules in parallel

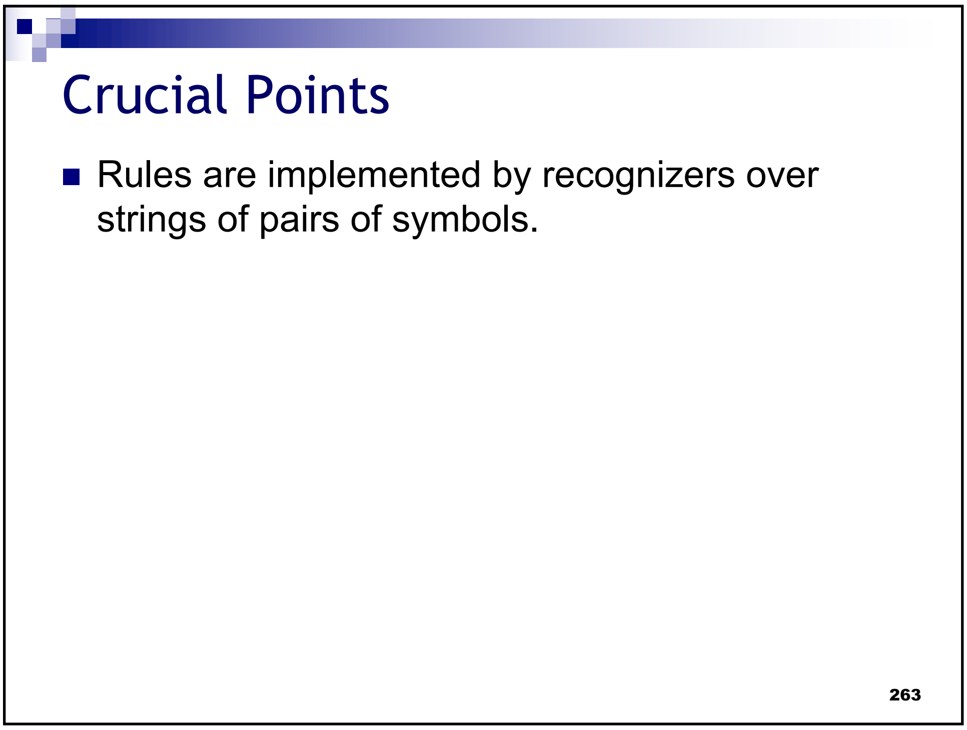


Both machines are now in accepting states

Rules in parallel

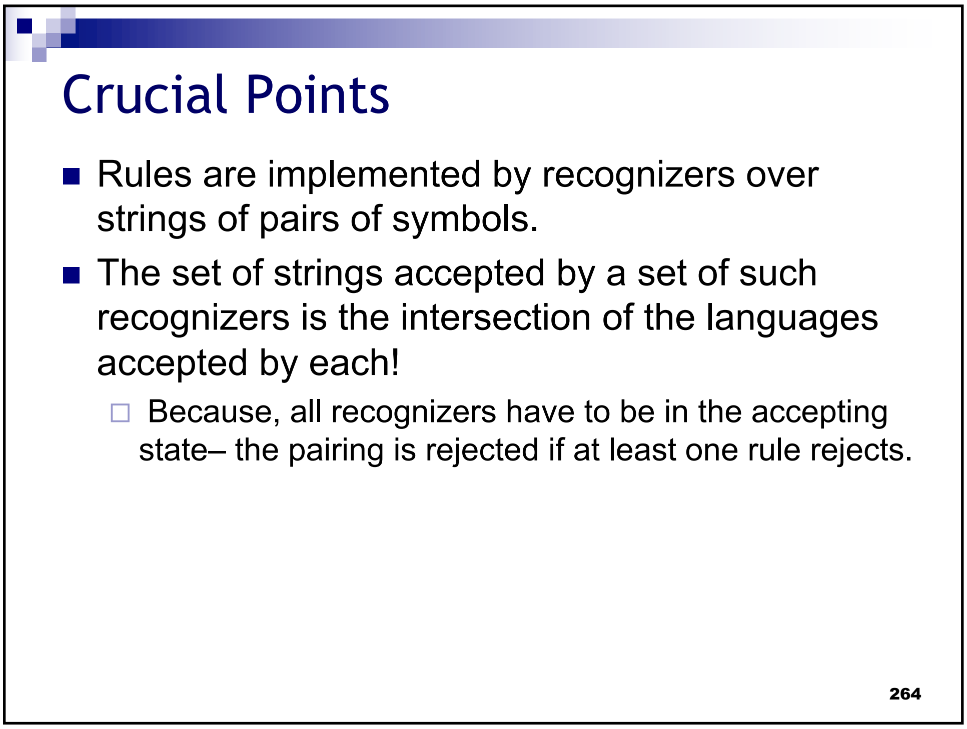


Try kaNpat and kampat pairing



Crucial Points

- Rules are implemented by recognizers over strings of pairs of symbols.



Crucial Points

- Rules are implemented by recognizers over strings of pairs of symbols.
- The set of strings accepted by a set of such recognizers is the intersection of the languages accepted by each!
  - □ Because, all recognizers have to be in the accepting state– the pairing is rejected if at least one rule rejects.

Crucial Points

- Rules are implemented by recognizers over strings of pairs of symbols.
- The set of strings accepted by a set of such recognizers is the intersection of the languages accepted by each!
- Such recognizers can be viewed as transducers between lexical and surface string languages.

Crucial Points

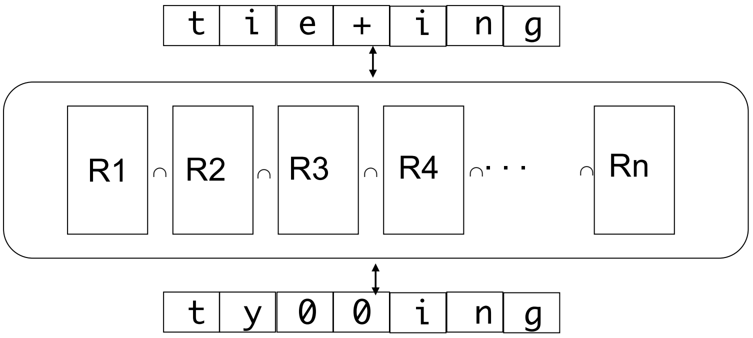
- Rules are implemented by recognizers over strings of pairs of symbols.
- The set of strings accepted by a set of such recognizers is the intersection of the languages accepted by each!
- Such machines can be viewed as transducers between lexical and surface string languages.
- Each transducer defines a regular relation or transduction.

Some Technical Details

- Such machines can be viewed as transducers between lexical and surface string languages.
- Each transducer defines a regular relation or transduction.
- The intersection of the regular relations over equal length strings is a regular relation (Kaplan and Kay 1994).
    - □ In general the intersection of regular relations need not be regular!
- So, 0 is treated as a normal symbol internally, not as ε. So there are no transitions with real ε symbols.

Intersecting the Transducers

- The transducers for each rule can now be intersected to give a single transducer.



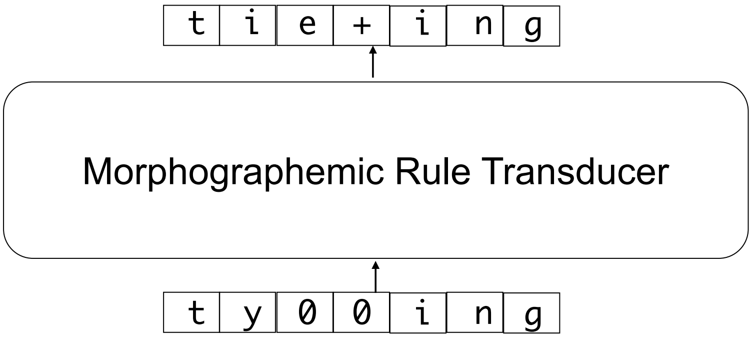
t i e + i n g

R1 ∩ R2 ∩ R3 ∩ R4 ∩· · · ∩ Rn

t y 0 0 i n g

The Intersected Transducer

- The resulting transducer can be used for analysis



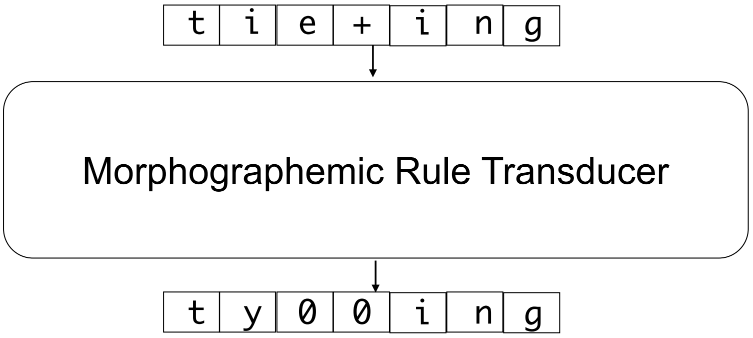
t i e + i n g

Morphographemic Rule Transducer

t y 0 0 i n g

The Intersected Transducer

- The resulting transducer can be used for analysis, or generation



t i e + i n g

Morphographemic Rule Transducer

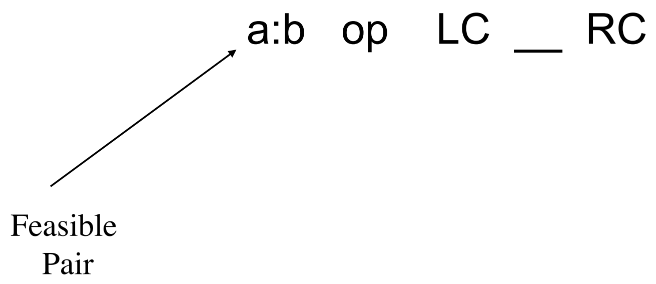
t y 0 0 i n g

Describing Phenomena

- Finite state transducers are too low level.
- Need high level notational tools for describing morphographemic phenomena
- Computers can then compile such rules into transducers; compilation is too much of a hassle for humans!

Two-level Rules

- Always remember the set of feasible symbols = sets of legal correspondences.
- Rules are of the sort:

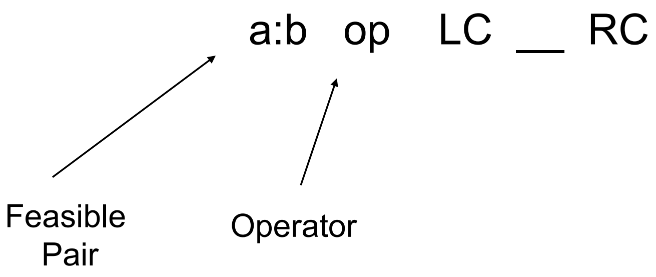


a:b op LC __ RC

Feasible Pair

Two-level Rules

- Rules are of the sort:



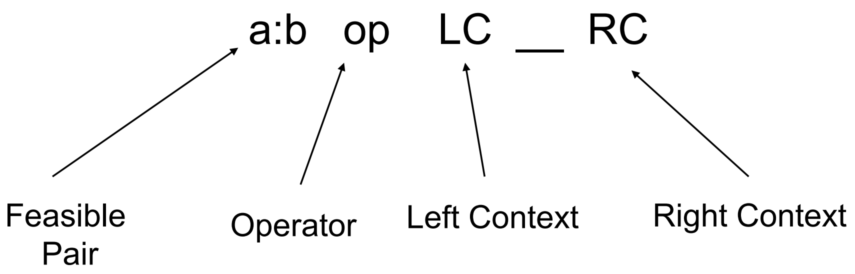
a:b op LC __ RC

Feasible Pair

Operator

Two-level Rules

- Rules are of the sort:



a:b op LC __ RC

Feasible Pair

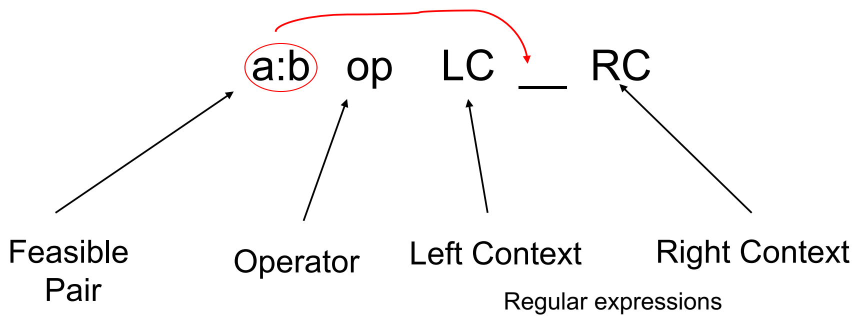
Operator

Left Context

Right Context

Two-level Rules

- Rules are of the sort:



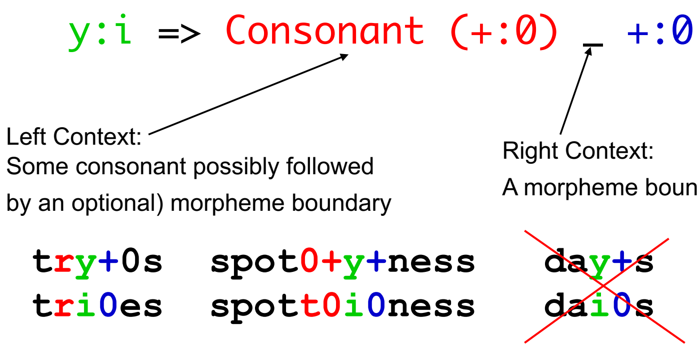
Regular expressions
that define what comes before
and after the pair.

The Context Restriction Rule

- `a:b    =>   LC _ RC;`
  - □ a lexical a MAY be paired with a surface b only in this context
  - □ correspondence implies the context
  - □ occurrence of a:b in any other context is not allowed (but you can specify multiple contexts!)

The Context Restriction Rule

a:b => LC _ RC;



y:i => Consonant (+:0) _ +:0

Left Context:
Some consonant possibly followed
by an optional) morpheme boundary

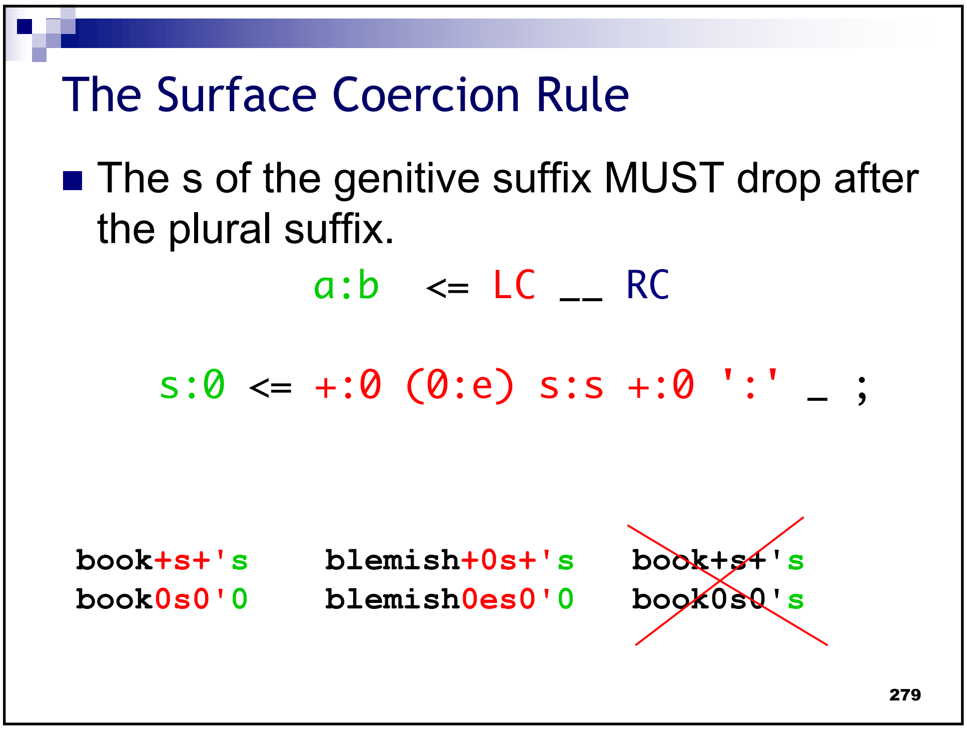
Right Context:
A morpheme boundary

```
try+0s    spot0+y+ness    ~~day+s~~
tri0es    spott0i0ness    ~~dai0s~~
```

The Surface Coercion Rule

- a:b <= LC _ RC

  - □ A lexical a MUST be paired with a surface b in the given context; no other pairs with a as its lexical symbol can appear in this context.
  - □ Context implies correspondence
  - □ Note that there may be other contexts where an a may be paired with b (optionally)



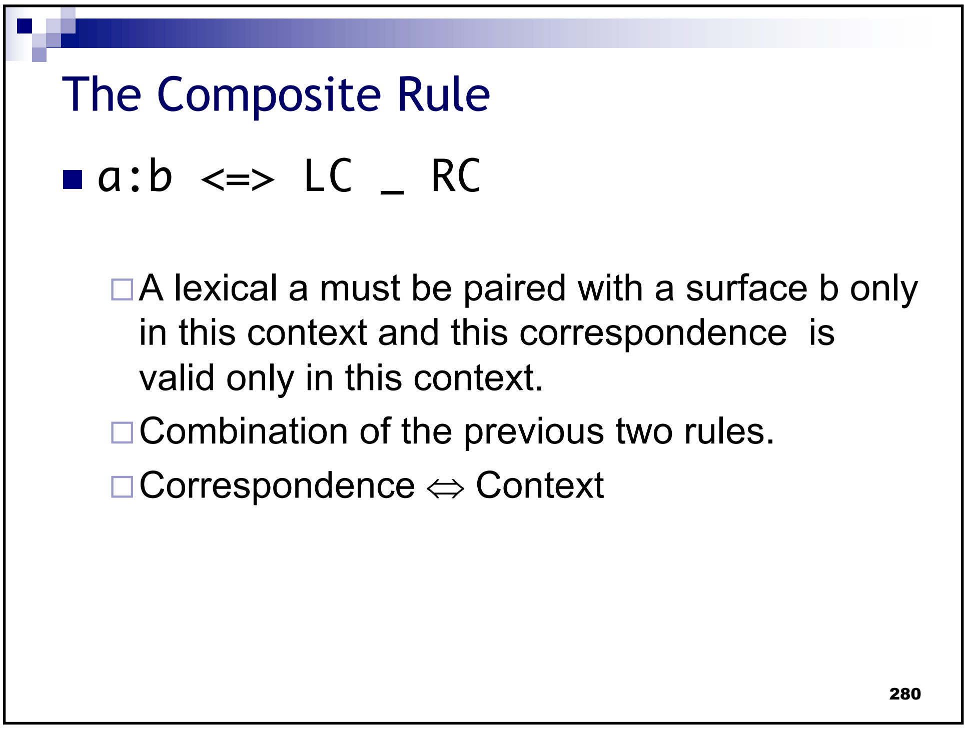
The Surface Coercion Rule

- The s of the genitive suffix MUST drop after the plural suffix.

`a:b <= LC __ RC`

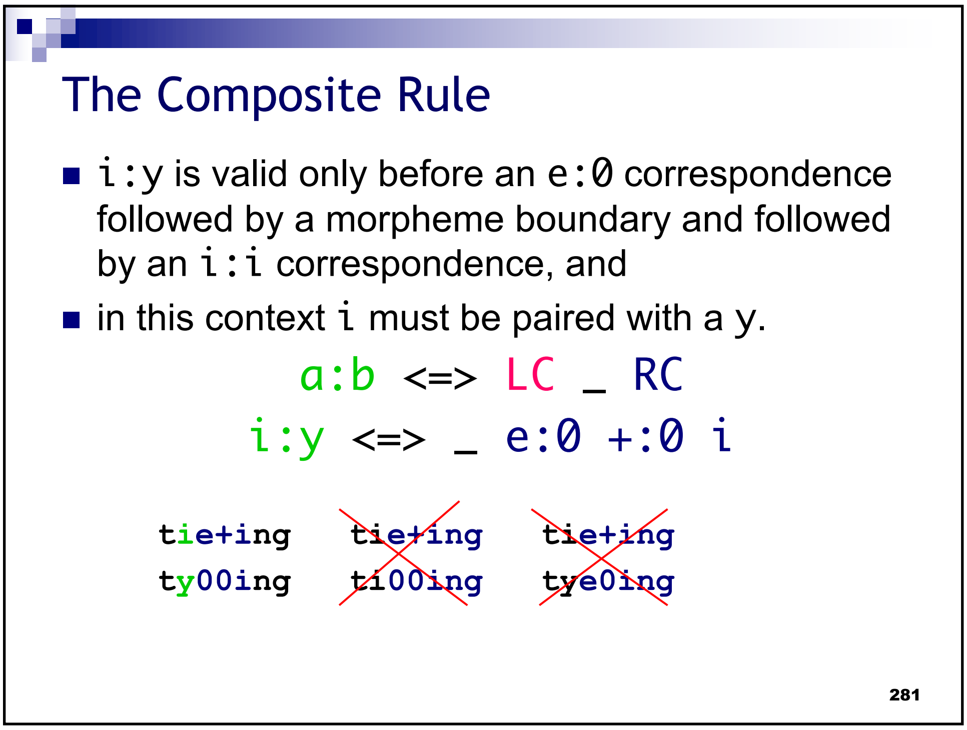
`s:0 <= +:0 (0:e) s:s +:0 ':' _ ;`

book+s+'s book0s0'0	blemish+0s+'s blemish0es0'0	~~book+s+'s~~ ~~book0s0's~~



The Composite Rule

- a:b <=> LC _ RC
  - □ A lexical a must be paired with a surface b only in this context and this correspondence is valid only in this context.
  - □ Combination of the previous two rules.
  - □ Correspondence ⇔ Context

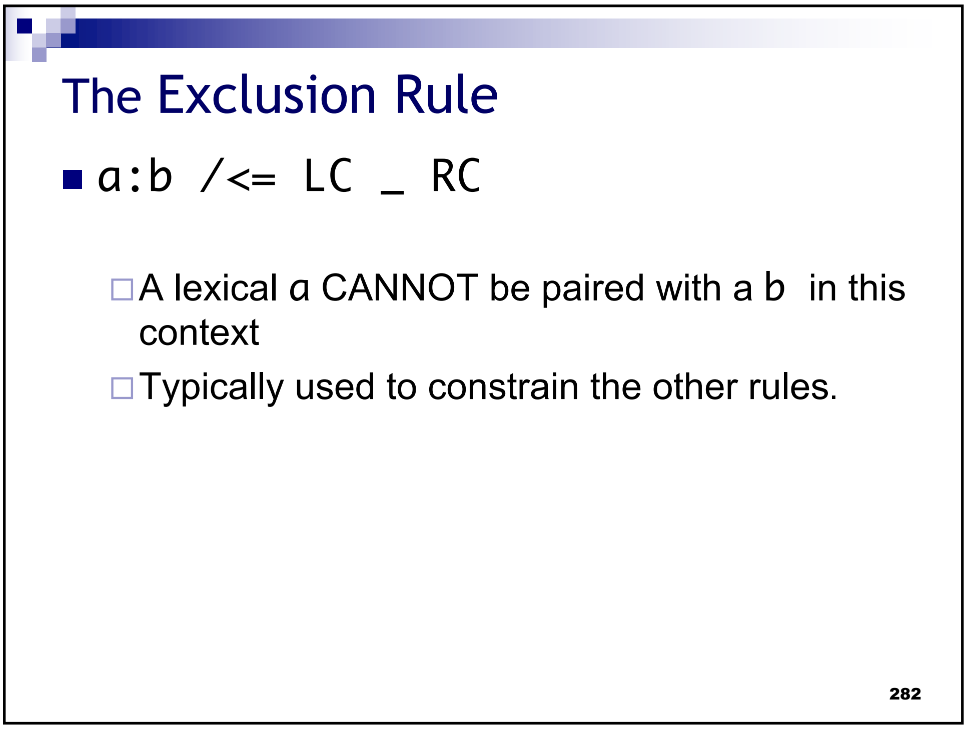


The Composite Rule

- `i:y` is valid only before an `e:0` correspondence followed by a morpheme boundary and followed by an `i:i` correspondence, and
- in this context `i` must be paired with a `y`.

```
a:b <=> LC _ RC
i:y <=> _ e:0 +:0 i
```

tie+ing	~~tie+ing~~	~~tie+ing~~
ty00ing	~~ti00ing~~	~~tye0ing~~



The Exclusion Rule

- `a:b /<= LC _ RC`
  - □ A lexical *a* CANNOT be paired with a b in this context
  - □ Typically used to constrain the other rules.

The Exclusion Rule

- The `y:i` correspondence formalized earlier can not occur if the morpheme on the right hand side starts with an `i` or the `'` (the genitive marker)

```
a:b /<= LC _ RC
y:i /<= Consonant (+:0) _ +:0 [i:i |
                    ’:’]
```

try+ing   ~~try+ing~~
try0ing   ~~tri+ing~~ ← Note that the previous context restriction rule sanctions this case

Rules to Transducers

- All the rule types can be compiled into finite state transducers
  - □Rather hairy and not so gratifying (☺)

Rules to Transducers

- Let’s think about a:b => LC _ RC
- If we see the a:b pair we want to make sure
  - It is preceded by a (sub)string that matches LC, and
  - It is followed by a (sub)string that matches RC
- So we reject any input that violates either or both of these constraints

Rules to Transducers

- Let’s think about a:b => LC _ RC
- More formally
  - It is not the case that we have a:b not preceded by LC, or not followed by RC
  - ~[

```
            [~ [?* LC ] a:b  ?*]  |
            [ ~ ?* a:b  ~[ RC ?* ] ]
  ]
```

(~ is the complementation operator)

Summary of Rules

- **<=　　a:b <= c _ d**
    - □ **a** is **always** realized as **b** in the context **c _ d**.
- **=>　　a:b => c _ d**
    - □ **a** is realized as **b only** in the context **c _ d**.
- **<=>　　a:b <=> c _ d**
    - □ **a** is realized as **b** in **c _ d** and nowhere else.
- **/<=　　a:b /<= c _ d**
    - □ **a** is **never** realized as **b** in the context **c _ d**.

How does one select a rule?

	Is a:b allowed in this context?	Is a:b only allowed in this context?	Must a always correspond to b in this context?
a:b => LC _ RC	Yes	Yes	No
a:b <= LC _ RC	Yes	No	Yes
a:b <=> LC_ RC	Yes	Yes	Yes
a:b /<= LC _ RC	No	NA	NA

More Rule Examples - kaNpat

- N:m <=> _ p:@;
  - □N:n is also given as a feasible pair.
- p:m <=> @:m _ ;

More Rule Examples

- A voiced consonant is devoiced word-finally
  - □b:p <=> _ #;
  - □d:t <=> _ #;
  - □c:ç <=> _ #;
- Xrce formalism lest you write this as a single rule
- **Cx:Cy <=> _ #: ;**

where Cx in (b d c)

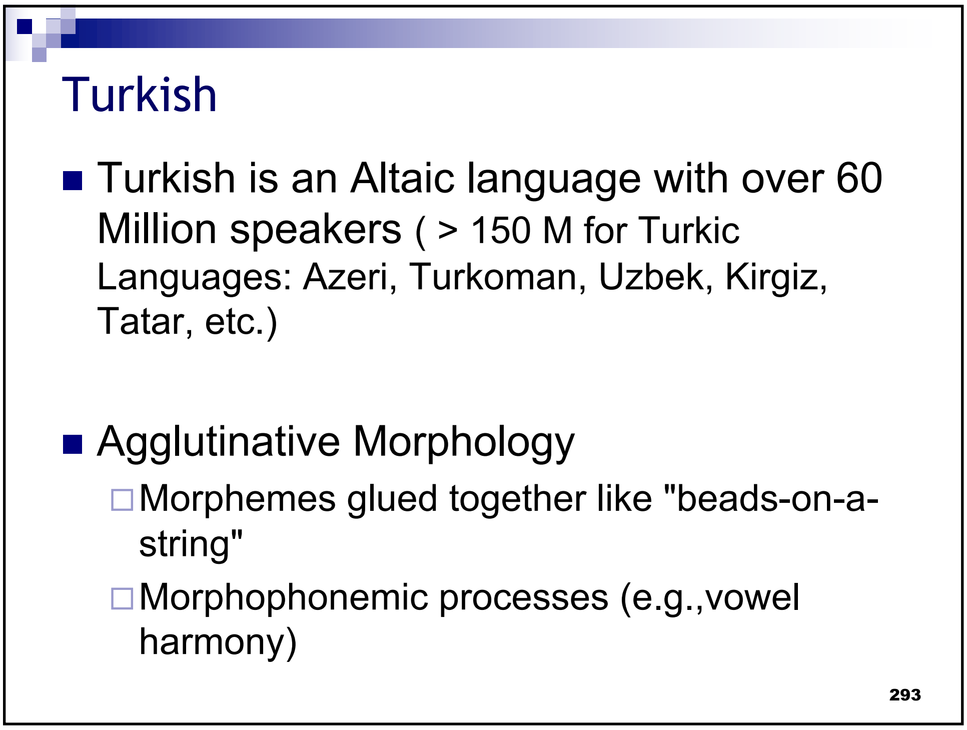
Cy in (p t ç) matched ;

Two-level Morphology

- Beesley and Karttunen, Finite State Morphology, CSLI Publications, 2004 (www.fsmbook.com)
- Karttunen and Beesley: Two-level rule compiler, Xerox PARC Tech Report
- Sproat, Morphology and Computation, MIT Press
- Ritchie et al. Computational Morphology, MIT Press
- Two-Level Rule Compiler

  http://www.xrce.xerox.com/competencies/content-analysis/fssoft/docs/twolc-92/twolc92.html

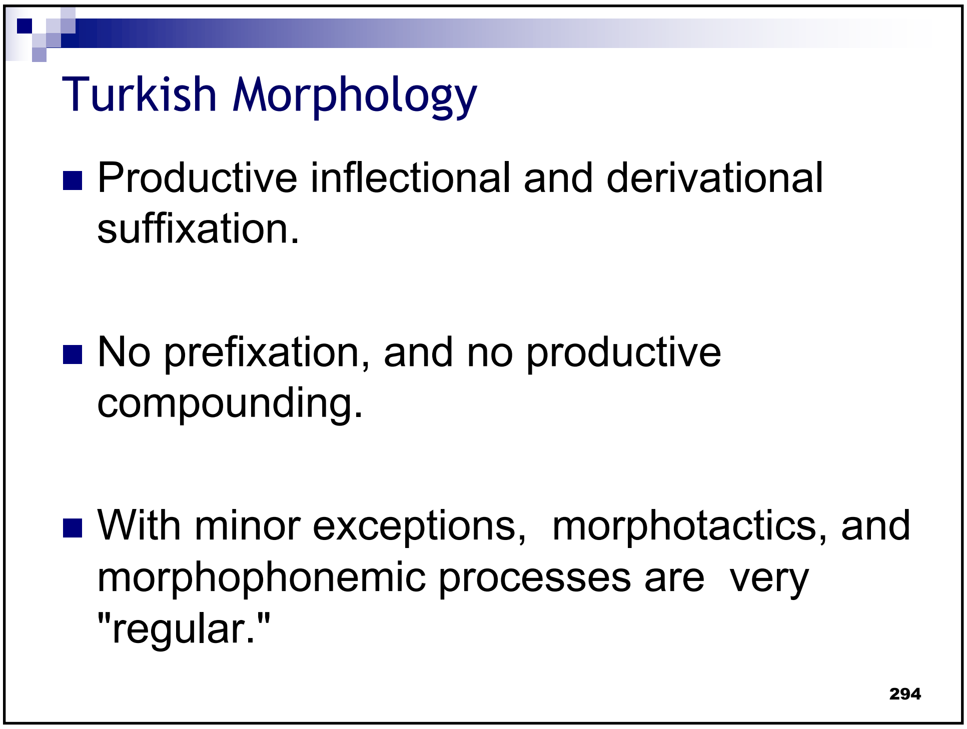
Engineering a Real Morphological Analyzer



Turkish

- Turkish is an Altaic language with over 60 Million speakers (> 150 M for Turkic Languages: Azeri, Turkoman, Uzbek, Kirgiz, Tatar, etc.)

- Agglutinative Morphology
  - □Morphemes glued together like "beads-on-a-string"
  - □Morphophonemic processes (e.g.,vowel harmony)



Turkish Morphology

- Productive inflectional and derivational suffixation.

- No prefixation, and no productive compounding.

- With minor exceptions, morphotactics, and morphophonemic processes are very "regular."

Turkish Morphology

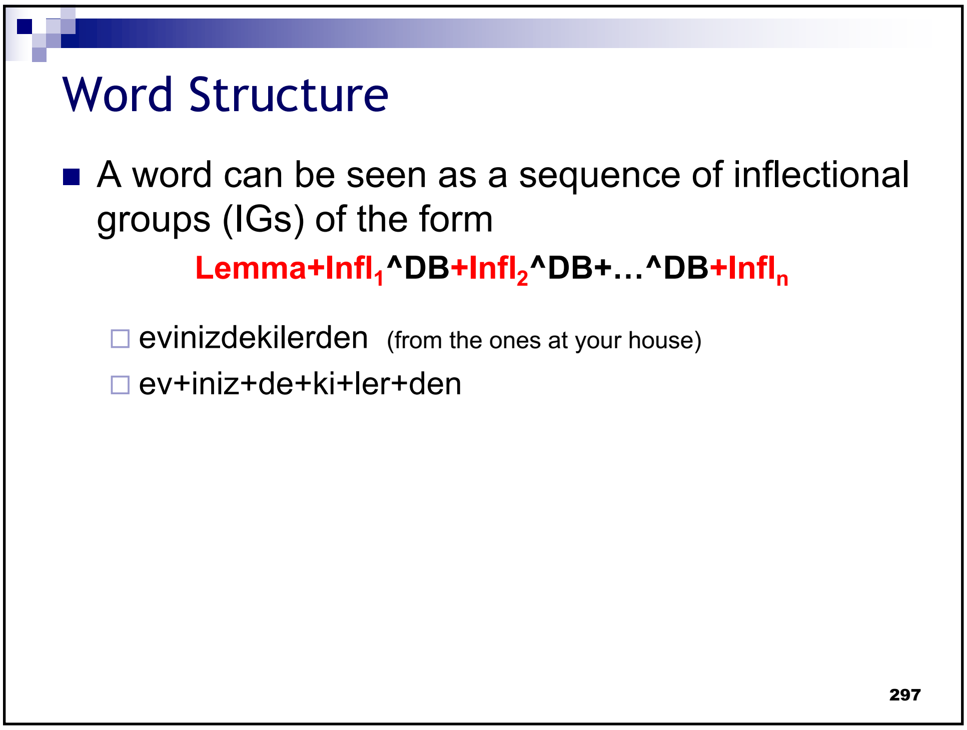
- Too many word forms per root.
  - □ Hankamer (1989) e.g., estimates few million forms per verbal root (based on generative capacity of derivations).
  - □ Nouns have about 100 different forms w/o any derivations
  - □ Verbs have a thousands.

Word Structure

- A word can be seen as a sequence of inflectional groups (IGs) of the form

  Lemma+Infl$_1$^DB+Infl$_2$^DB+…^DB+Infl$_n$

  - □ evinizdekilerden (from the ones at your house)

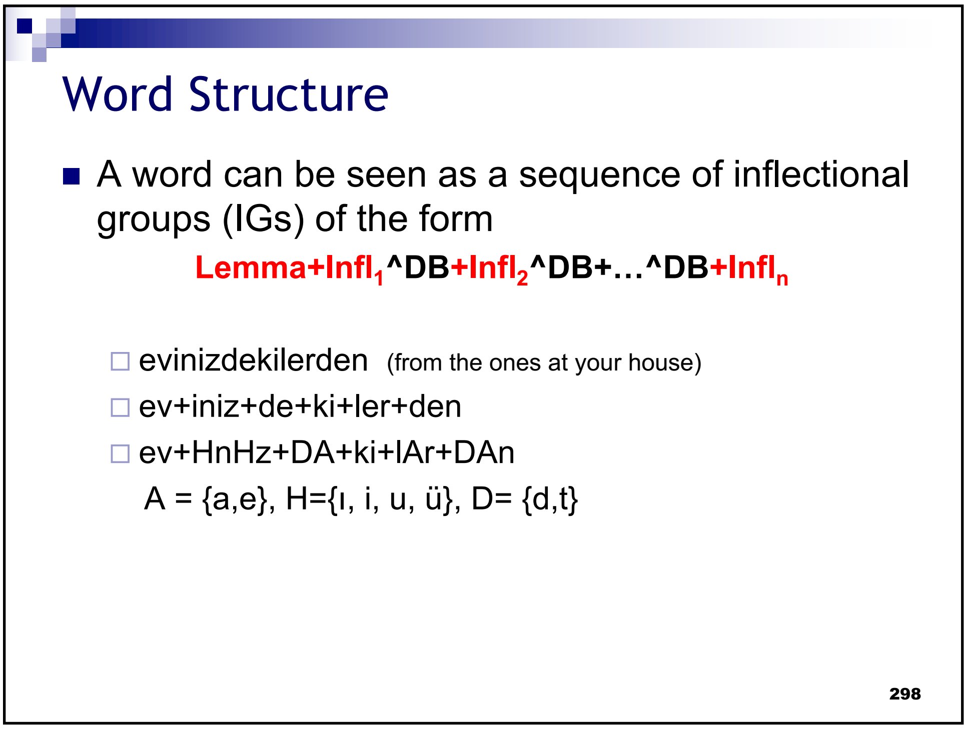


Word Structure

- A word can be seen as a sequence of inflectional groups (IGs) of the form

  Lemma+Infl$_1$^DB+Infl$_2$^DB+…^DB+Infl$_n$

  - □ evinizdekilerden (from the ones at your house)
  - □ ev+iniz+de+ki+ler+den



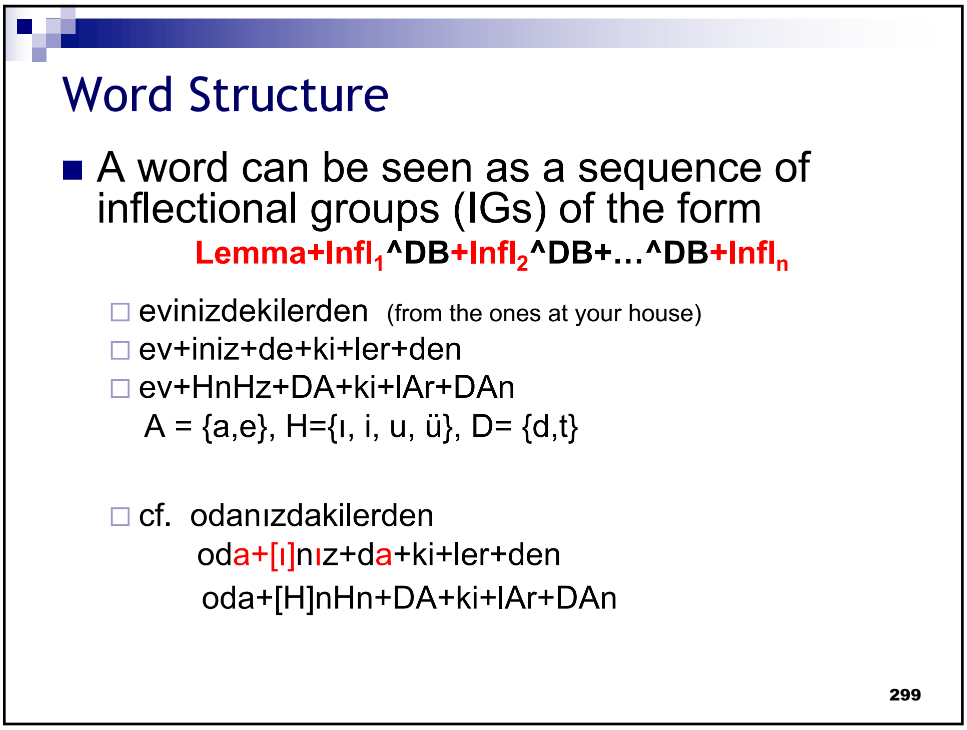
Word Structure

- A word can be seen as a sequence of inflectional groups (IGs) of the form

  Lemma+Infl$_1$^DB+Infl$_2$^DB+…^DB+Infl$_n$

  - □ evinizdekilerden (from the ones at your house)
  - □ ev+iniz+de+ki+ler+den
  - □ ev+HnHz+DA+ki+lAr+DAn

    A = {a,e}, H={ı, i, u, ü}, D= {d,t}



Word Structure

- A word can be seen as a sequence of inflectional groups (IGs) of the form

  Lemma+Infl$_1$^DB+Infl$_2$^DB+…^DB+Infl$_n$

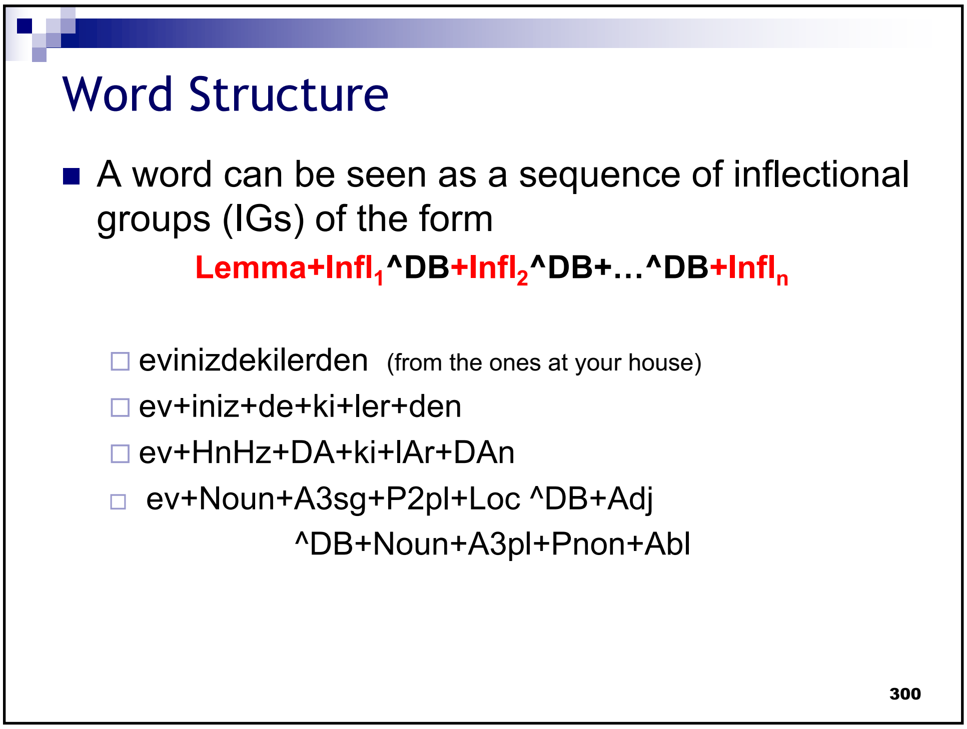
  - □ evinizdekilerden (from the ones at your house)
  - □ ev+iniz+de+ki+ler+den
  - □ ev+HnHz+DA+ki+lAr+DAn

    A = {a,e}, H={ı, i, u, ü}, D= {d,t}

  - □ cf. odanızdakilerden

    oda+[ı]nız+da+ki+ler+den

    oda+[H]nHn+DA+ki+lAr+DAn



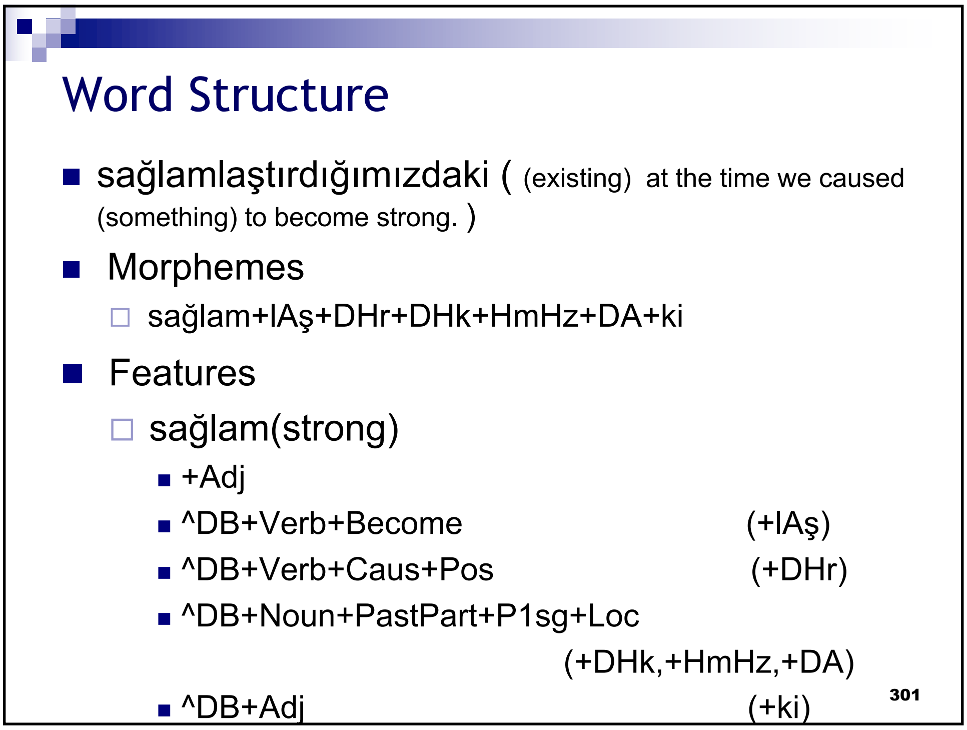
Word Structure

- A word can be seen as a sequence of inflectional groups (IGs) of the form

  Lemma+Infl$_1$^DB+Infl$_2$^DB+…^DB+Infl$_n$

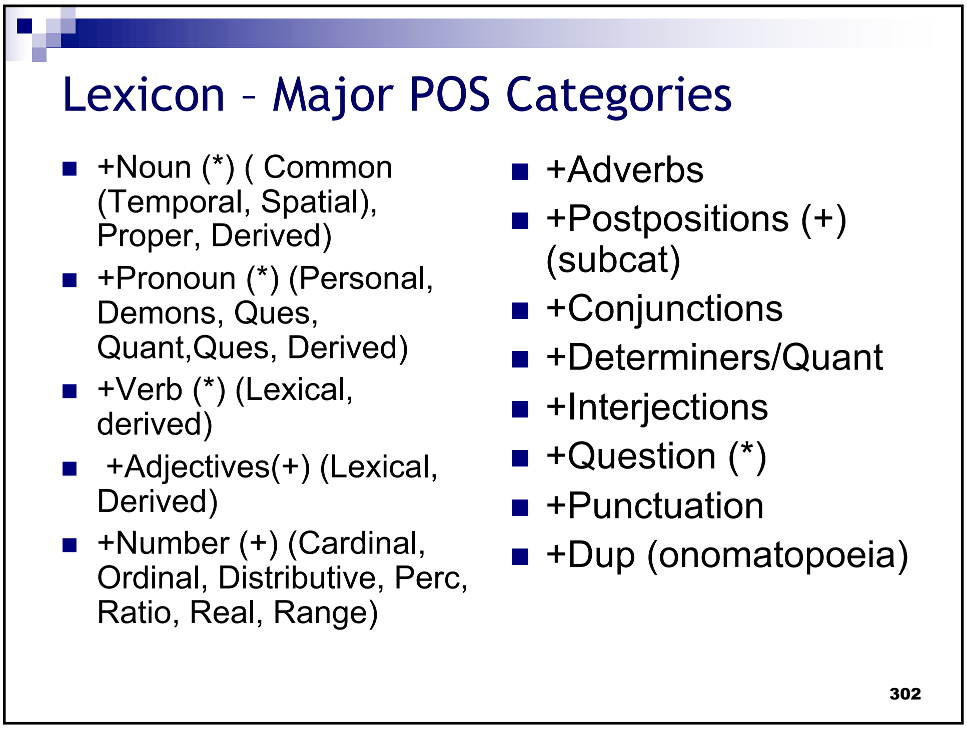
  - □ evinizdekilerden (from the ones at your house)
  - □ ev+iniz+de+ki+ler+den
  - □ ev+HnHz+DA+ki+lAr+DAn
  - □ ev+Noun+A3sg+P2pl+Loc ^DB+Adj

    ^DB+Noun+A3pl+Pnon+Abl



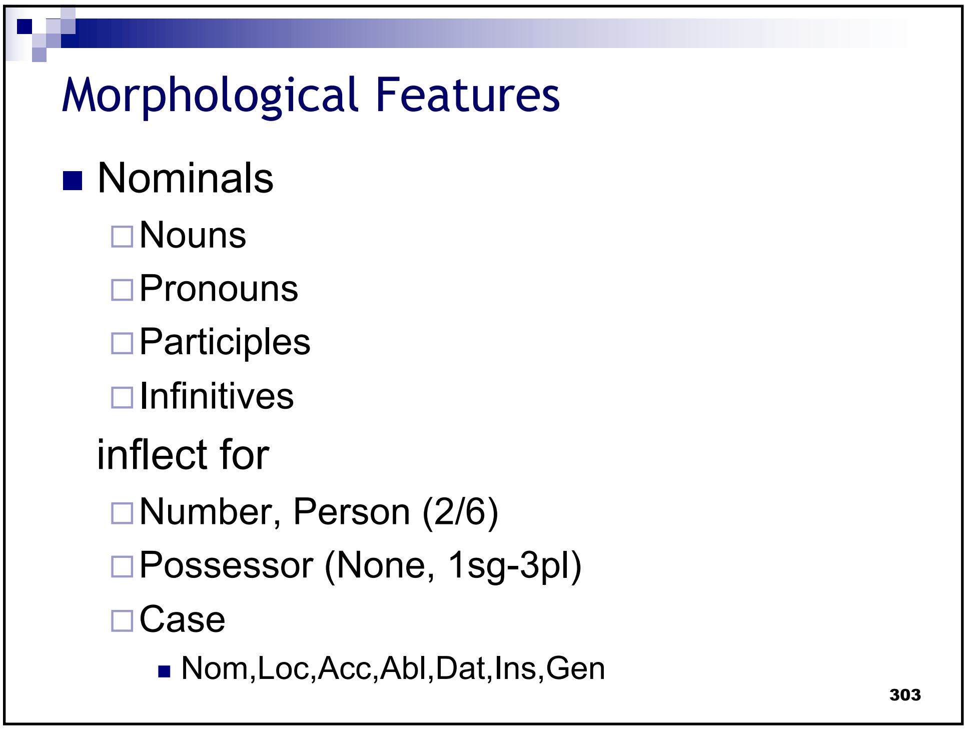
Word Structure

- sağlamlaştırdığımızdaki ((existing) at the time we caused (something) to become strong.)
- Morphemes
  - □ sağlam+lAş+DHr+DHk+HmHz+DA+ki
- Features
  - □ sağlam(strong)
    - +Adj
    - ^DB+Verb+Become (+lAş)
    - ^DB+Verb+Caus+Pos (+DHr)
    - ^DB+Noun+PastPart+P1sg+Loc (+DHk,+HmHz,+DA)
    - ^DB+Adj (+ki)



Lexicon – Major POS Categories

- +Noun (*) (Common (Temporal, Spatial), Proper, Derived)
- +Pronoun (*) (Personal, Demons, Ques, Quant,Ques, Derived)
- +Verb (*) (Lexical, derived)
- +Adjectives(+) (Lexical, Derived)
- +Number (+) (Cardinal, Ordinal, Distributive, Perc, Ratio, Real, Range)
- +Adverbs
- +Postpositions (+) (subcat)
- +Conjunctions
- +Determiners/Quant
- +Interjections
- +Question (*)
- +Punctuation
- +Dup (onomatopoeia)

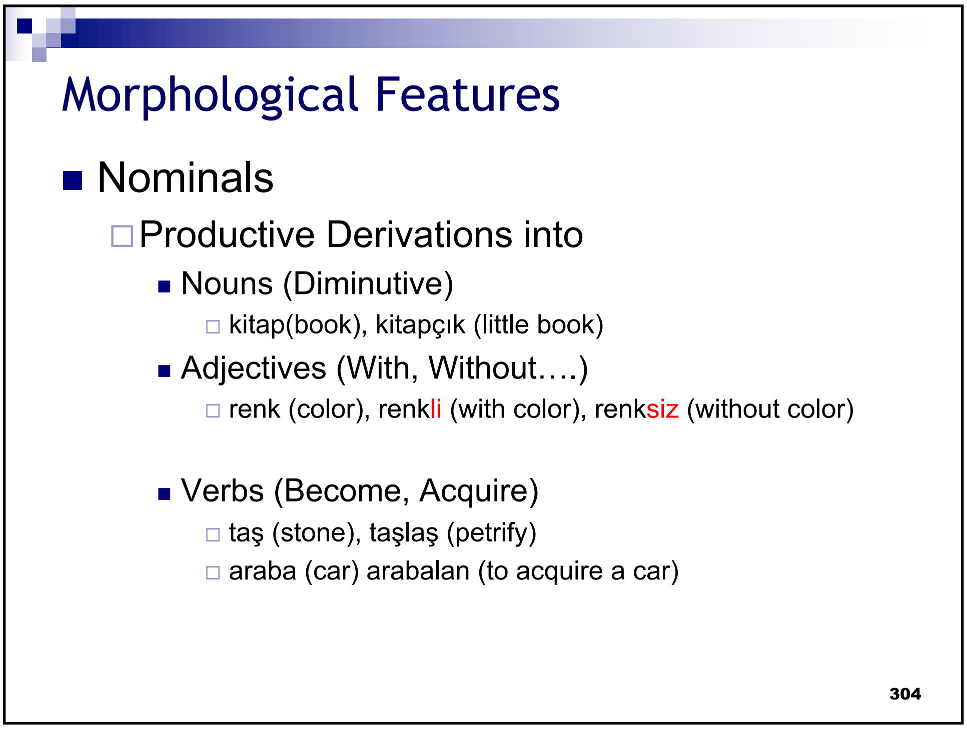


Morphological Features

- Nominals
  - □ Nouns
  - □ Pronouns
  - □ Participles
  - □ Infinitives

  inflect for
  - □ Number, Person (2/6)
  - □ Possessor (None, 1sg-3pl)
  - □ Case
    - Nom,Loc,Acc,Abl,Dat,Ins,Gen

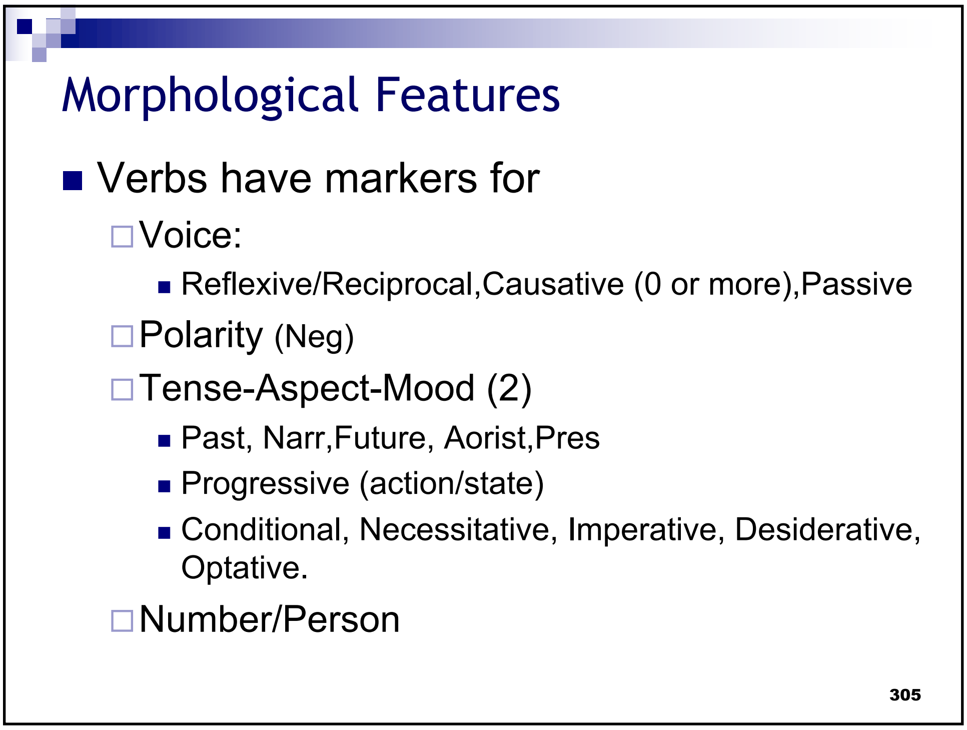
303



Morphological Features

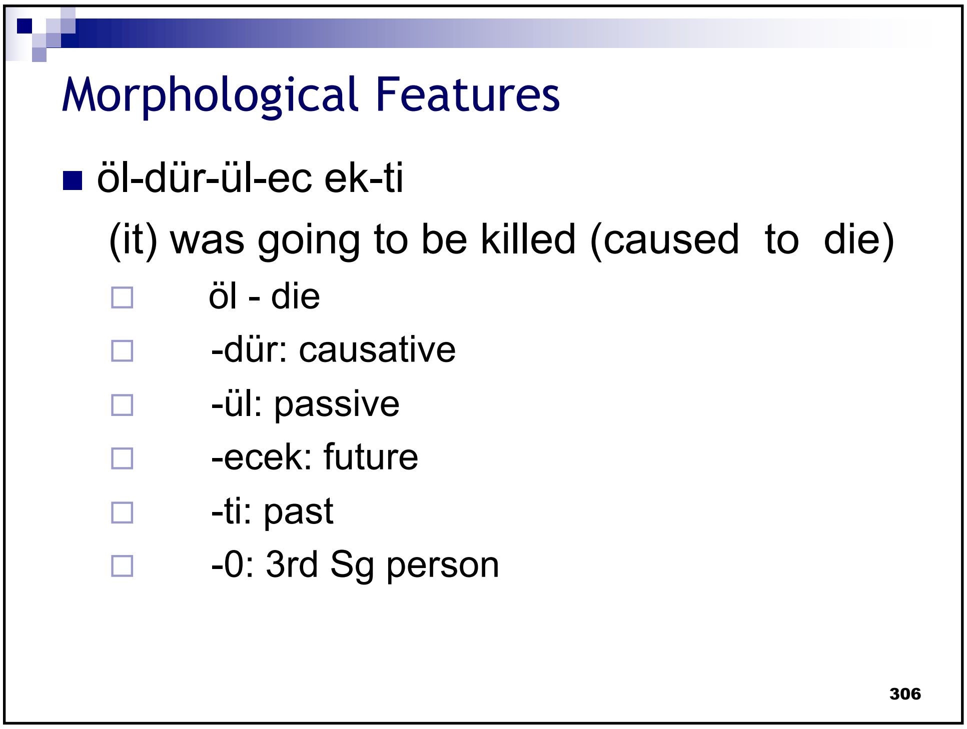
- Nominals
  - □ Productive Derivations into
    - Nouns (Diminutive)
      - □ kitap(book), kitapçık (little book)
    - Adjectives (With, Without….)
      - □ renk (color), renkli (with color), renksiz (without color)
    - Verbs (Become, Acquire)
      - □ taş (stone), taşlaş (petrify)
      - □ araba (car) arabalan (to acquire a car)

304



Morphological Features

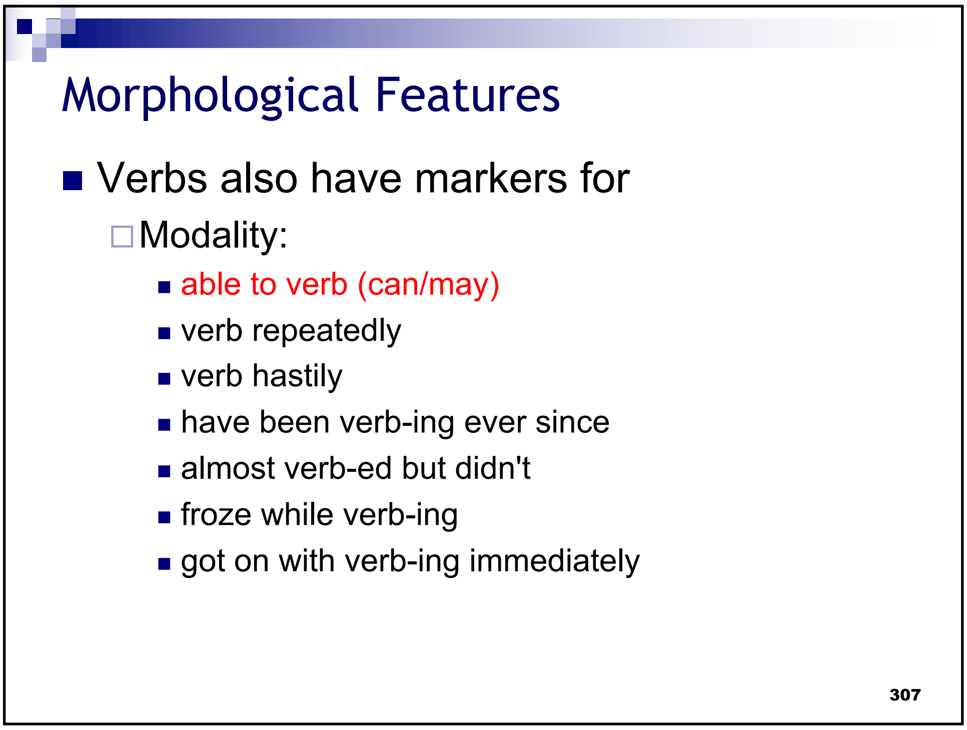
- Verbs have markers for
  - Voice:
    - Reflexive/Reciprocal,Causative (0 or more),Passive
  - Polarity (Neg)
  - Tense-Aspect-Mood (2)
    - Past, Narr,Future, Aorist,Pres
    - Progressive (action/state)
    - Conditional, Necessitative, Imperative, Desiderative, Optative.
  - Number/Person



Morphological Features

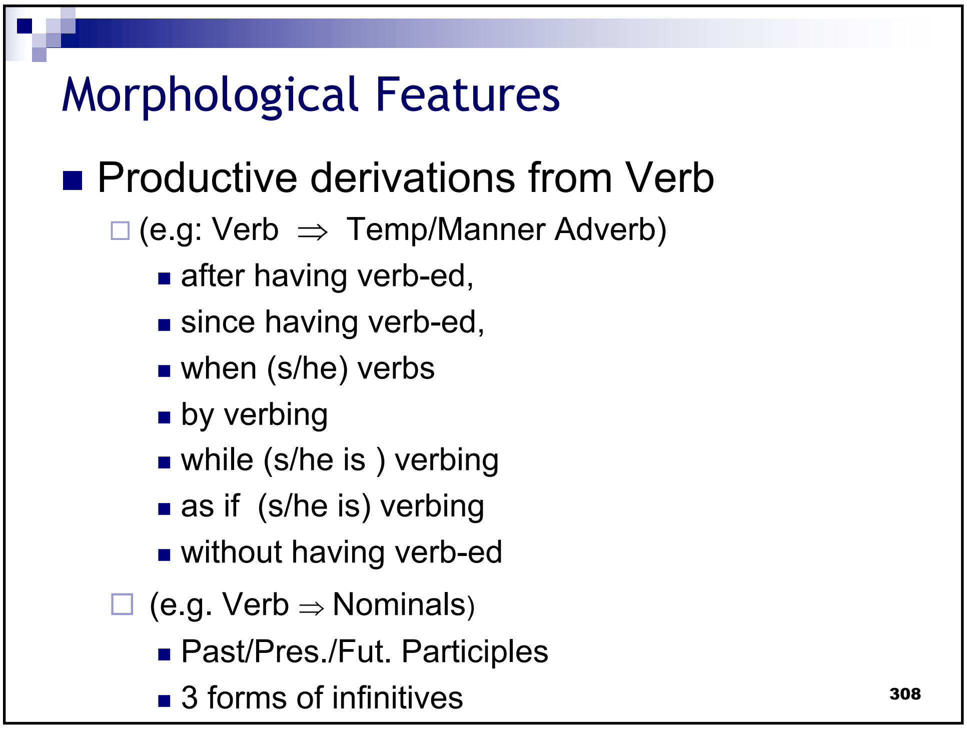
- öl-dür-ül-ec ek-ti

  (it) was going to be killed (caused to die)
  - öl - die
  - -dür: causative
  - -ül: passive
  - -ecek: future
  - -ti: past
  - -0: 3rd Sg person



Morphological Features

- Verbs also have markers for
  - □ Modality:
    - able to verb (can/may)
    - verb repeatedly
    - verb hastily
    - have been verb-ing ever since
    - almost verb-ed but didn't
    - froze while verb-ing
    - got on with verb-ing immediately



Morphological Features

- Productive derivations from Verb
  - □ (e.g: Verb ⇒ Temp/Manner Adverb)
    - after having verb-ed,
    - since having verb-ed,
    - when (s/he) verbs
    - by verbing
    - while (s/he is) verbing
    - as if (s/he is) verbing
    - without having verb-ed
  - □ (e.g. Verb ⇒ Nominals)
    - Past/Pres./Fut. Participles
    - 3 forms of infinitives

Morphographemic Processes

- Vowel Harmony
  - Vowels in suffixes agree in certain phonological features with the preceding vowels.

High Vowels H = {ı, i, u, ü}
Low Vowels = {a, e, o, ö}
Front Vowels = {e, i, ö, ü}
Back Vowels = {a, ı, o, u}
Round Vowels = { o, ö, u, ü}
Nonround Vowels = {a, e, ı, i}
Nonround Low A= {a, e}

Morphemes use A and H as underspecified
meta symbols on the lexical side.
+lAr : Plural Marker
+nHn: Genitive Case Marker

309

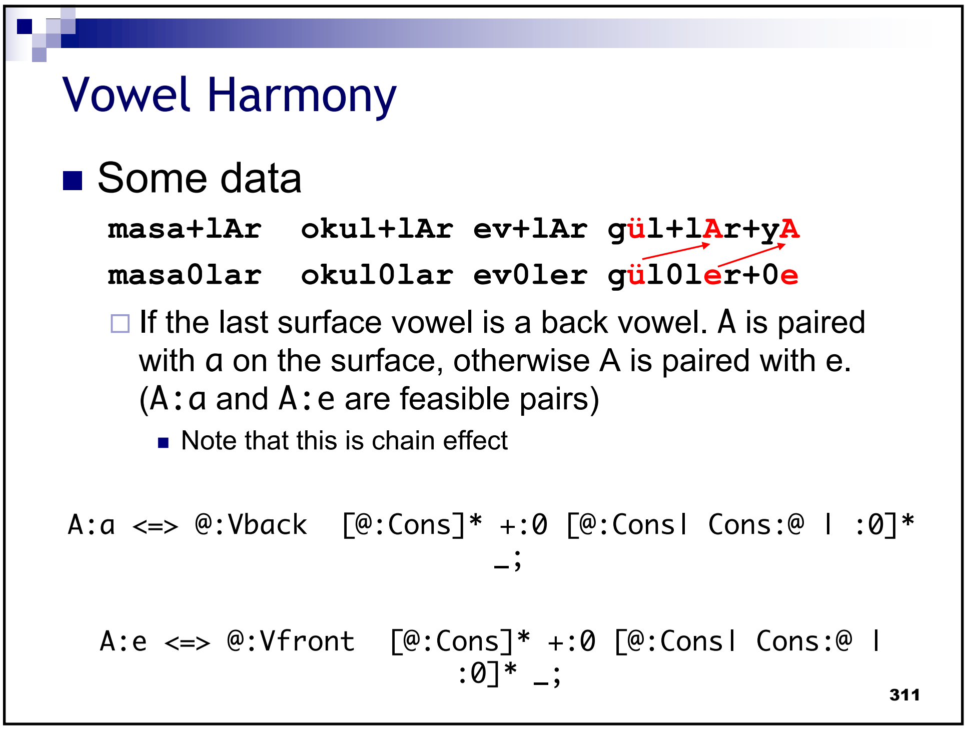
Vowel Harmony

- Some data

```
masa+lAr  okul+lAr ev+lAr gül+lAr
masa0lar  okul0lar ev0ler gül0ler
```

  - If the last surface vowel is a back vowel, A is paired with a on the surface, otherwise A is paired with e. (A:a and A:e are feasible pairs)

310



Vowel Harmony

- Some data

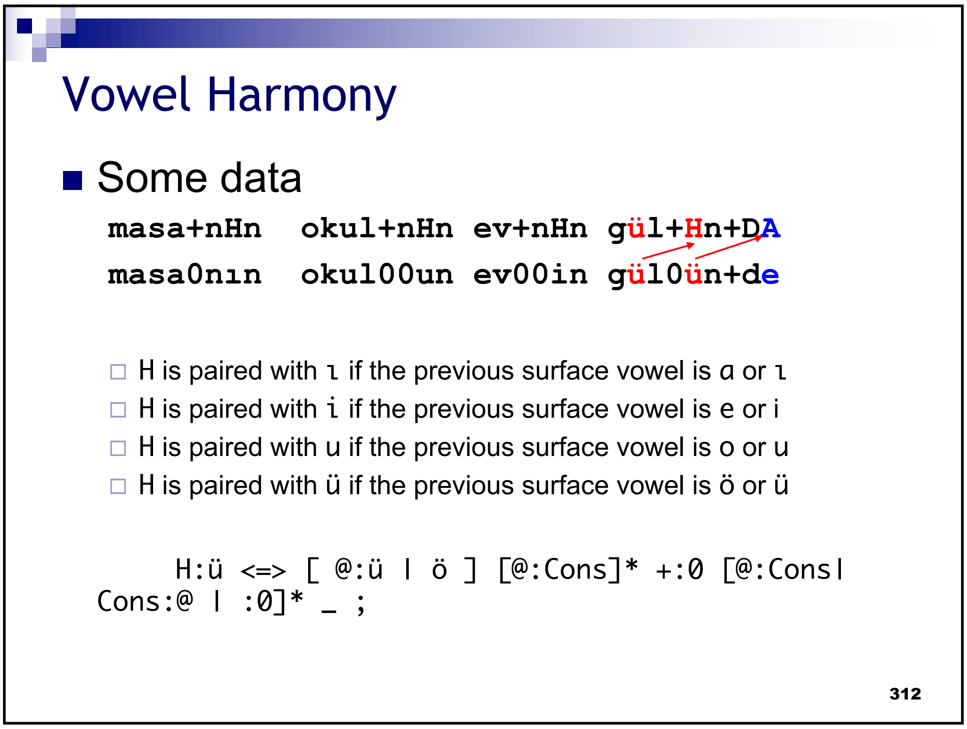
  masa+lAr okul+lAr ev+lAr gül+lAr+yA

  masa0lar okul0lar ev0ler gül0ler+0e

  - □ If the last surface vowel is a back vowel. A is paired with ɑ on the surface, otherwise A is paired with e. (A:ɑ and A:e are feasible pairs)
    - Note that this is chain effect

```
A:ɑ <=> @:Vback  [@:Cons]* +:0 [@:Cons| Cons:@ | :0]*
                         _;

  A:e <=> @:Vfront  [@:Cons]* +:0 [@:Cons| Cons:@ |
                        :0]* _;
```

311



Vowel Harmony

- Some data

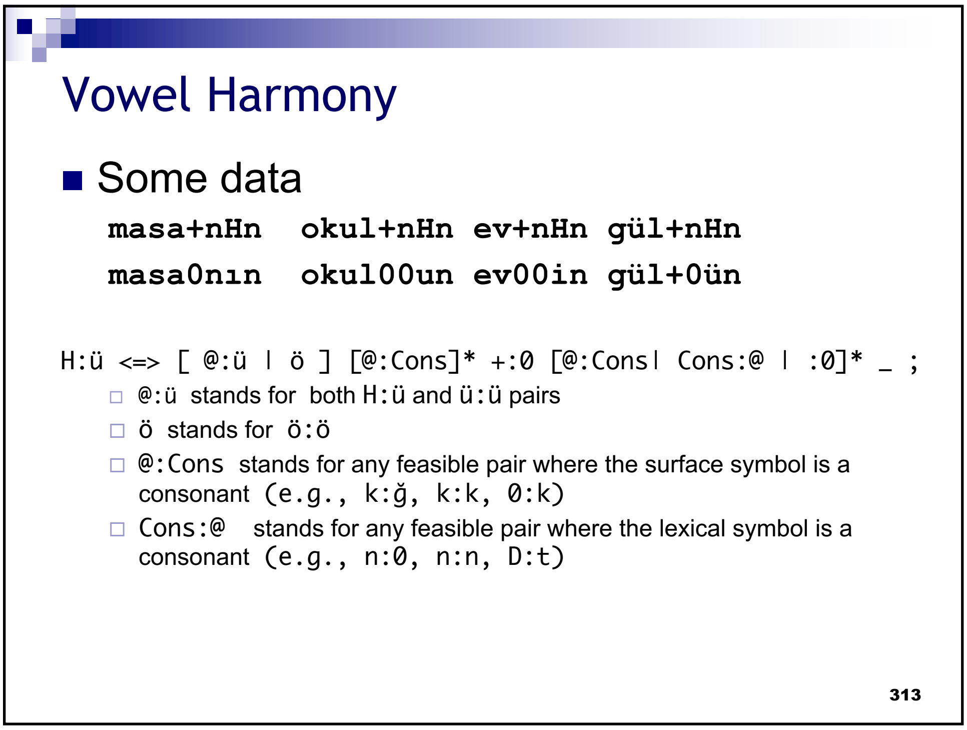
  masa+nHn okul+nHn ev+nHn gül+Hn+DA

  masa0nın okul00un ev00in gül0ün+de

  - □ H is paired with ı if the previous surface vowel is ɑ or ı
  - □ H is paired with i if the previous surface vowel is e or i
  - □ H is paired with u if the previous surface vowel is o or u
  - □ H is paired with ü if the previous surface vowel is ö or ü

```
     H:ü <=> [ @:ü | ö ] [@:Cons]* +:0 [@:Cons|
Cons:@ | :0]* _ ;
```

312



Vowel Harmony

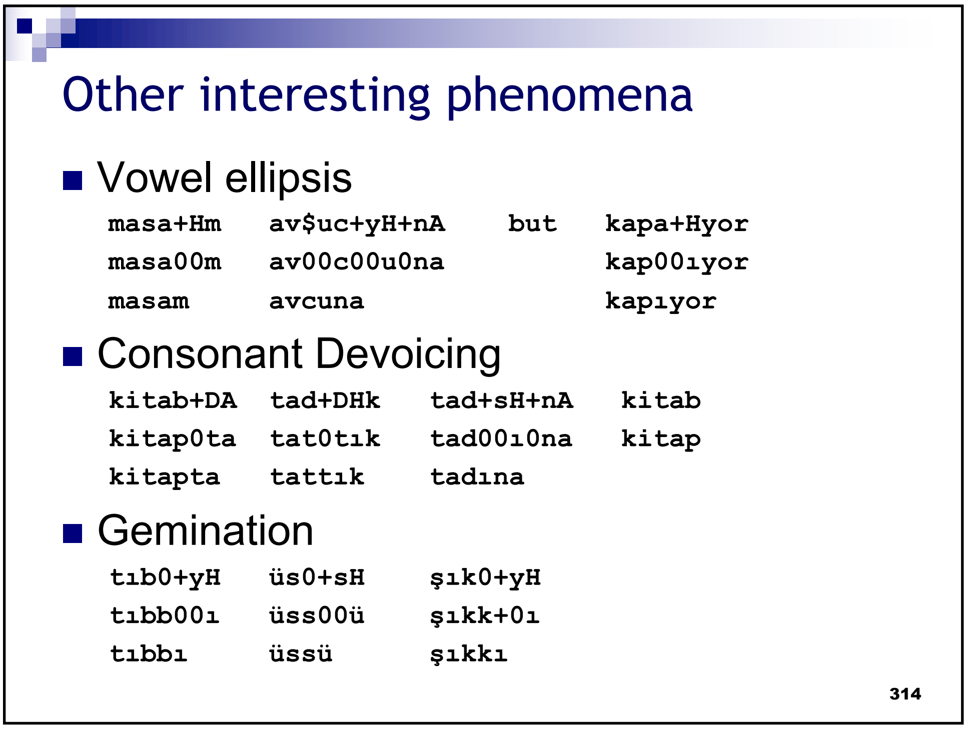
- Some data

masa+nHn	**okul+nHn**	**ev+nHn**	**gül+nHn**
masa0nın	**okul00un**	**ev00in**	**gül+0ün**

H:ü <=> [@:ü | ö] [@:Cons]* +:0 [@:Cons| Cons:@ | :0]* _ ;

- □ @:ü stands for both H:ü and ü:ü pairs
- □ ö stands for ö:ö
- □ @:Cons stands for any feasible pair where the surface symbol is a consonant (e.g., k:ğ, k:k, 0:k)
- □ Cons:@ stands for any feasible pair where the lexical symbol is a consonant (e.g., n:0, n:n, D:t)

313



Other interesting phenomena

- Vowel ellipsis

masa+Hm	**av$uc+yH+nA**	**but**	**kapa+Hyor**
masa00m	**av00c00u0na**		**kap00ıyor**
masam	**avcuna**		**kapıyor**

- Consonant Devoicing

kitab+DA	**tad+DHk**	**tad+sH+nA**	**kitab**
kitap0ta	**tat0tık**	**tad00ı0na**	**kitap**
kitapta	**tattık**	**tadına**	

- Gemination

tıb0+yH	**üs0+sH**	**şık0+yH**
tıbb00ı	**üss00ü**	**şıkk+0ı**
tıbbı	**üssü**	**şıkkı**

314

Other interesting phenomena

- Consonant ellipsis

```
ev+nHn    kalem+sH    ev+yH
ev00in    kalem00i    ev00i
```

- Other Consonant Changes

```
ayak+nHn renk+yH     radyolog+sH
ayağ00ın reng00i     radyoloğ00u

k:ğ <=> @:Vowel _ +:0  ([s:0|n:0|y:0]) @:Vowel
```

Reality Check-1

- Real text contains phenomena that causes nasty problems:
  - □ Words of foreign origin - 1

```
alkol+sH     kemal0+yA
alkol00ü     kemal'00e
```

    Use different lexical vowel symbols for these

  - □ Words of foreign origin -2

```
Carter'a     serverlar   Bordeaux'yu
```

    - This needs to be handled by a separate analyzer using phonological encodings of foreign words, or
    - Using Lenient morphology

Reality Check-2

- Real text contains phenomena that cause nasty problems:
  - □ Numbers, Numbers:

```
2'ye, 3'e, 4'ten, %6'dan, 20inci,100üncü
16:15 vs 3:4,  2/3'ü,  2/3'si
```

Affixation proceeds according to how the number is pronounced, not how it is written!

The number lexicon has to be coded so that a representation of the last bits of its pronunciation is available at the lexical level.

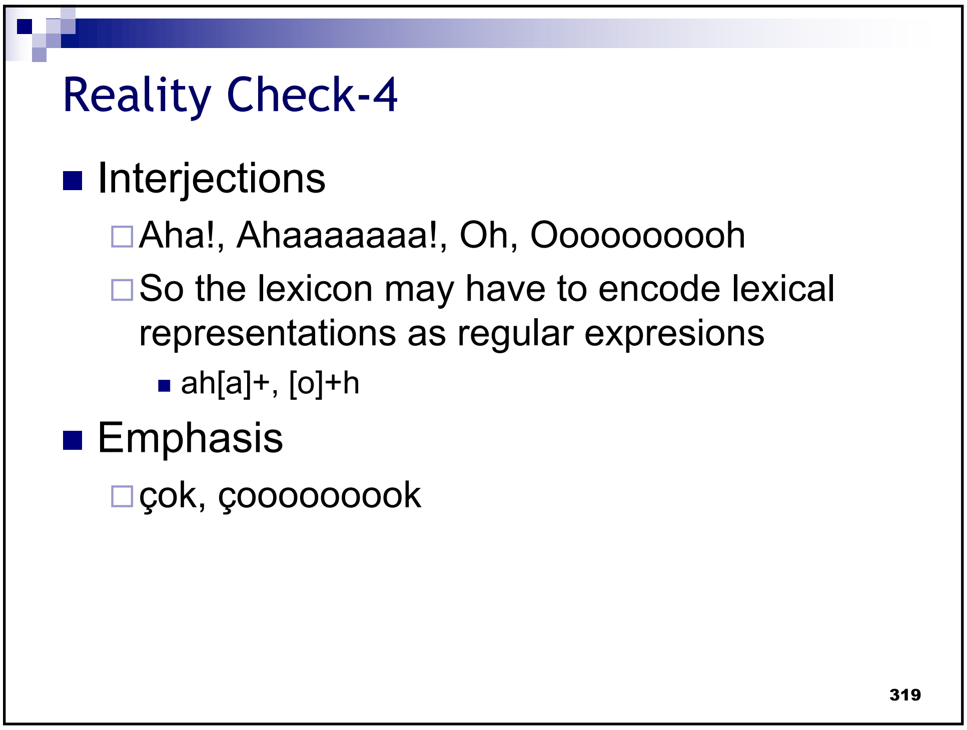
Reality Check-3

- Real text contains phenomena that causes nasty problems:
  - □ Acronyms

    PTTye -- No written vowel to harmonize to!

  - □ Stash an invisible symbol (E:0) into the lexical representation so that you can force harmonization

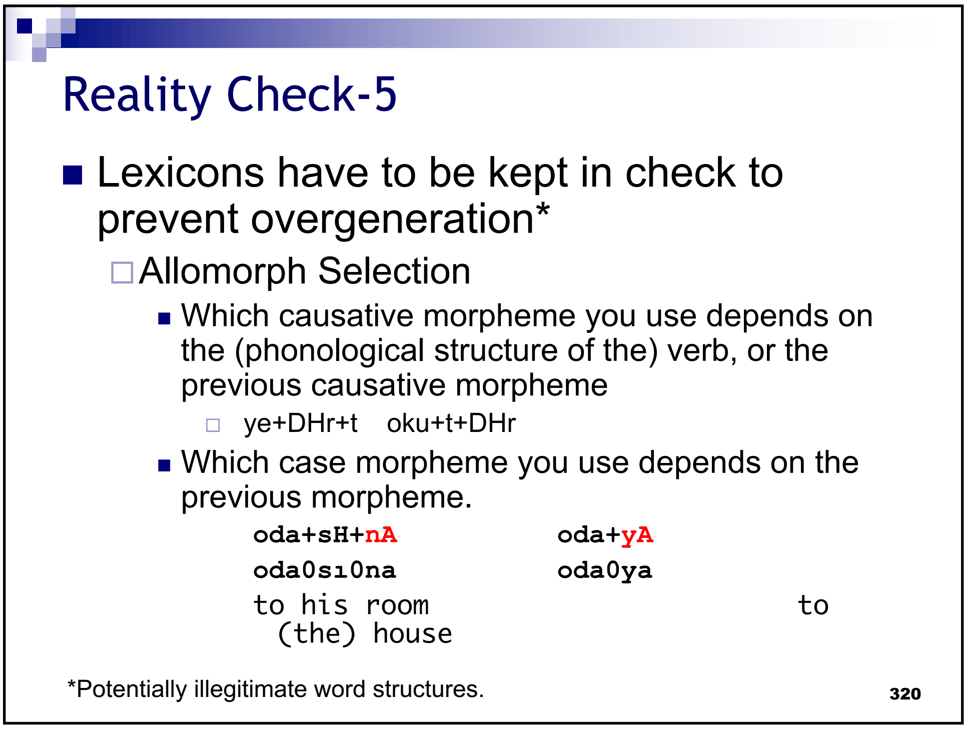
```
pttE+yA
ptt00ye
```



Reality Check-4

- Interjections
  - □ Aha!, Ahaaaaaaa!, Oh, Oooooooooh
  - □ So the lexicon may have to encode lexical representations as regular expresions
    - ah[a]+, [o]+h
- Emphasis
  - □ çok, çoooooooook

319



Reality Check-5

- Lexicons have to be kept in check to prevent overgeneration*
  - □ Allomorph Selection
    - Which causative morpheme you use depends on the (phonological structure of the) verb, or the previous causative morpheme
      - □ ye+DHr+t oku+t+DHr
    - Which case morpheme you use depends on the previous morpheme.

oda+sH+nA	**oda+yA**
oda0sı0na	**oda0ya**
to his room	to (the) house

*Potentially illegitimate word structures.

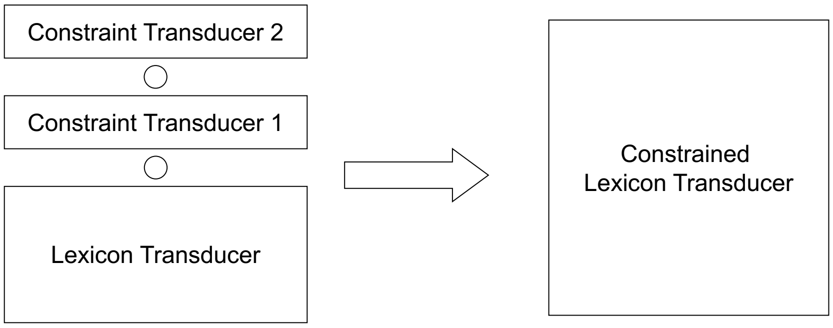
320

Reality Check-6

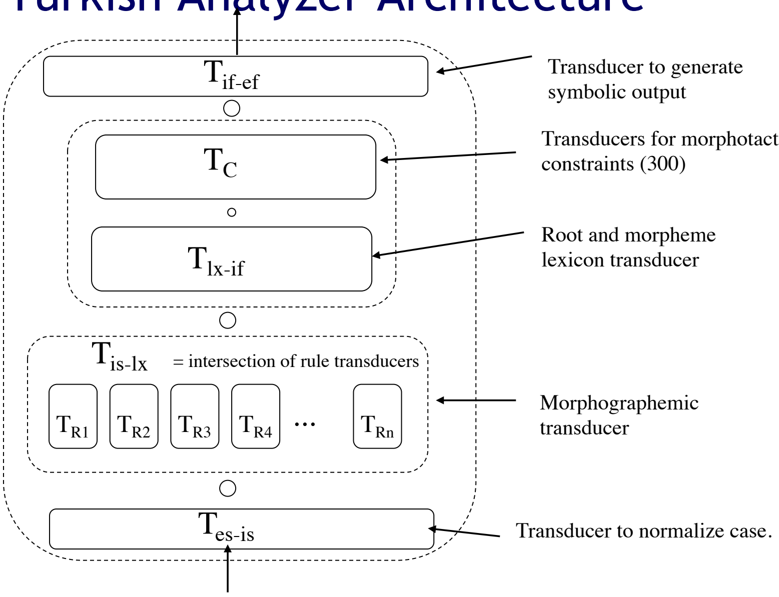
- Lexicons have to be kept in check to prevent overgeneration
  - +ki can only follow suffix only after +Loc case marked nouns, or
  - after singular nouns in +Nom case denoting temporal entities (such as day, minute, etc)

Taming Overgeneration

- All these can be specified as finite state transducers.



Turkish Analyzer Architecture



Transducer to generate symbolic output

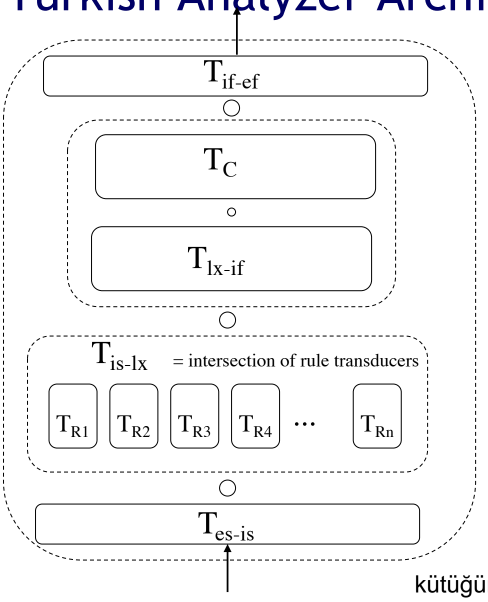
Transducers for morphotactic constraints (300)

Root and morpheme lexicon transducer

Morphographemic transducer

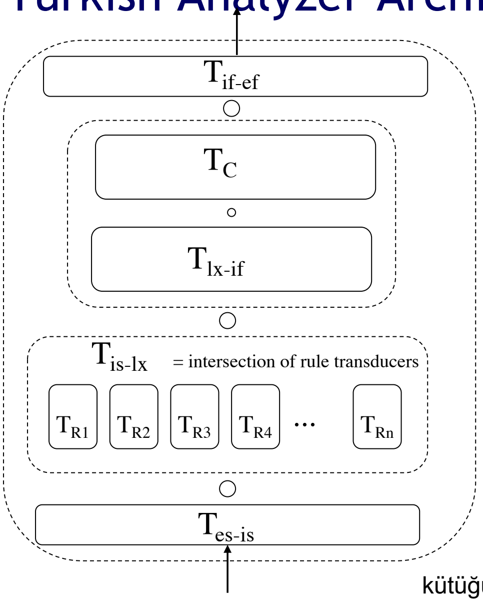
Transducer to normalize case.

Turkish Analyzer Architecture



kütüğünden, Kütüğünden, KÜTÜĞÜNDEN

Turkish Analyzer Architecture

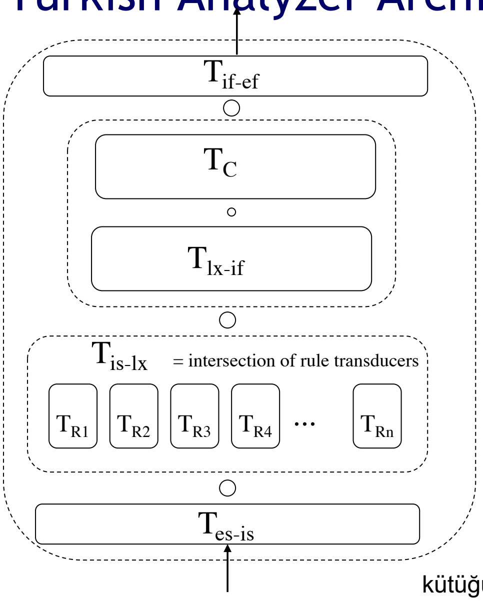


kütüğünden

↑

kütüğünden, Kütüğünden, KÜTÜĞÜNDEN

Turkish Analyzer Architecture



kütük+sH+ndAn

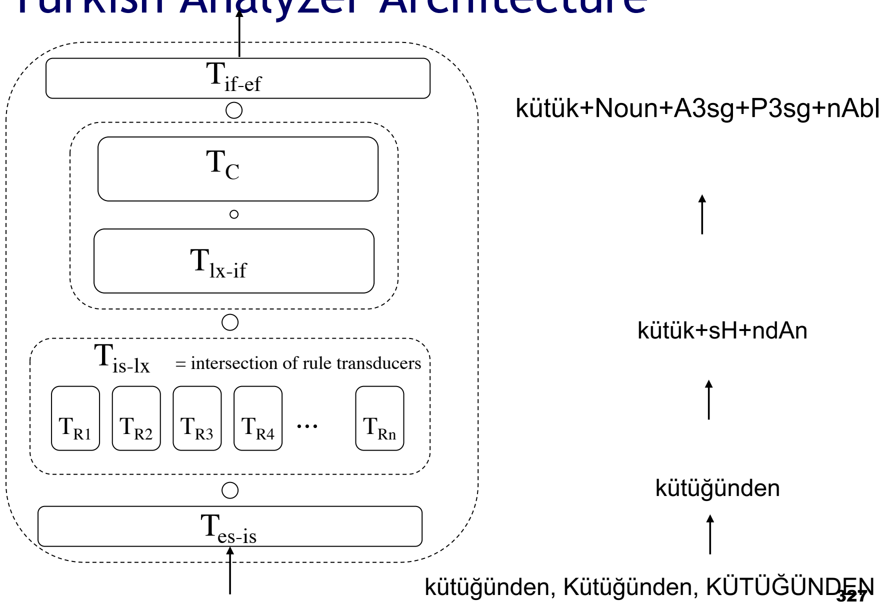
↑

kütüğünden

↑

kütüğünden, Kütüğünden, KÜTÜĞÜNDEN

Turkish Analyzer Architecture



$T_{if\text{-}ef}$

○

T_C

∘

$T_{lx\text{-}if}$

○

$T_{is\text{-}lx}$ = intersection of rule transducers

T_{R1} T_{R2} T_{R3} T_{R4} … T_{Rn}

○

$T_{es\text{-}is}$

kütük+Noun+A3sg+P3sg+nAbl

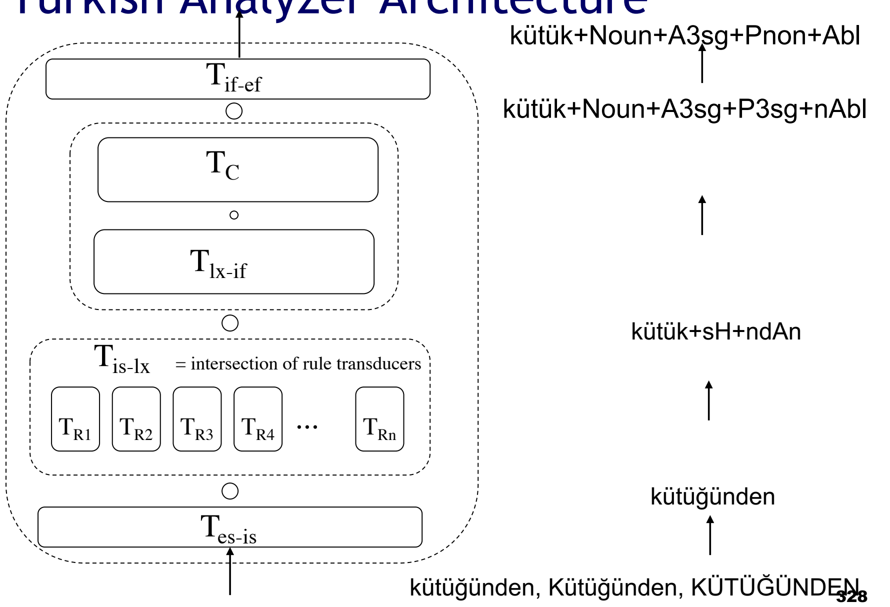
kütük+sH+ndAn

kütüğünden

kütüğünden, Kütüğünden, KÜTÜĞÜNDEN

327

Turkish Analyzer Architecture



$T_{if\text{-}ef}$

○

T_C

∘

$T_{lx\text{-}if}$

○

$T_{is\text{-}lx}$ = intersection of rule transducers

T_{R1} T_{R2} T_{R3} T_{R4} … T_{Rn}

○

$T_{es\text{-}is}$

kütük+Noun+A3sg+Pnon+Abl

kütük+Noun+A3sg+P3sg+nAbl

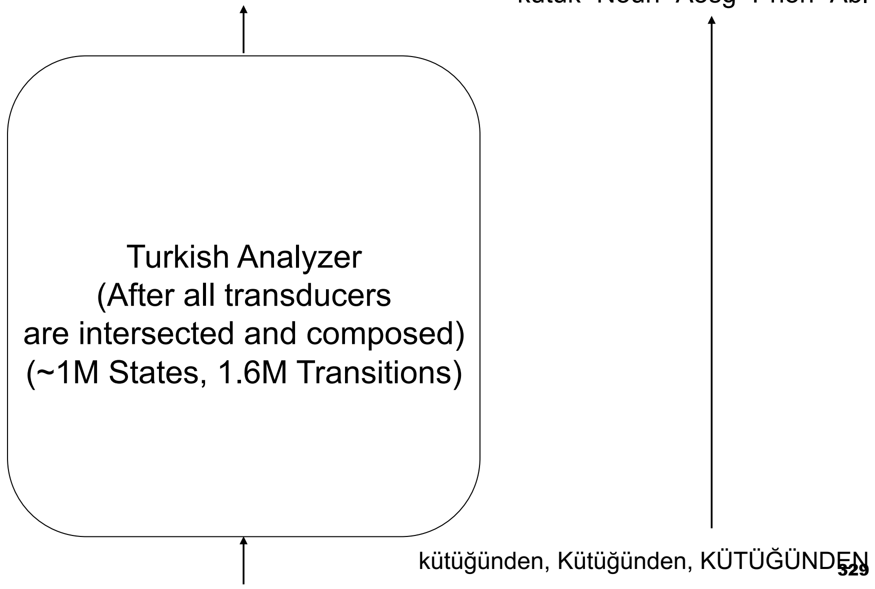
kütük+sH+ndAn

kütüğünden

kütüğünden, Kütüğünden, KÜTÜĞÜNDEN

328

Turkish Analyzer Architecture

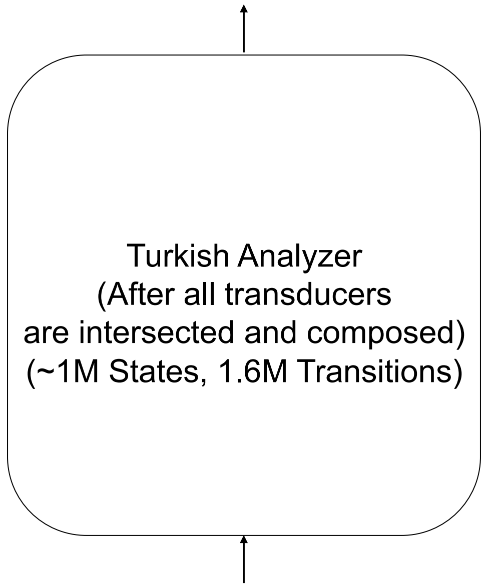


kütük+Noun+A3sg+Pnon+Abl

Turkish Analyzer
(After all transducers
are intersected and composed)
(~1M States, 1.6M Transitions)

kütüğünden, Kütüğünden, KÜTÜĞÜNDEN

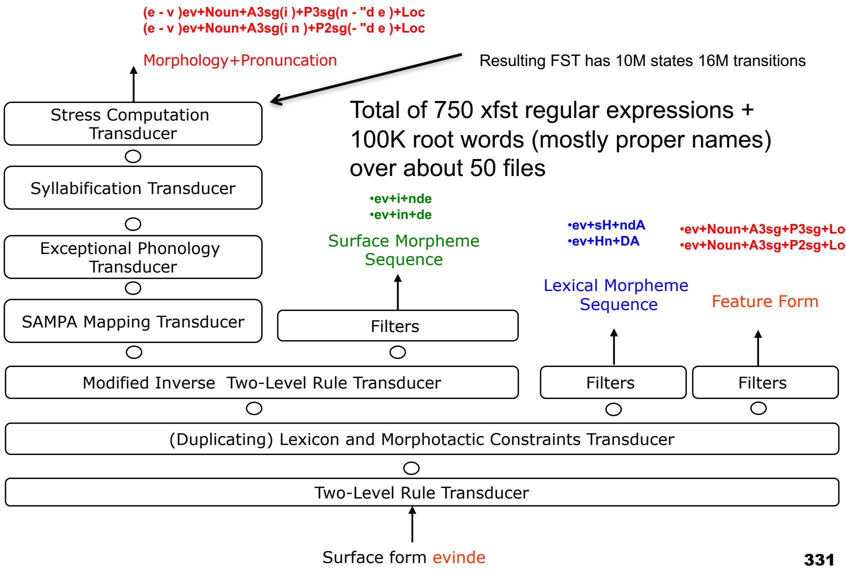
Turkish Analyzer Architecture



Turkish Analyzer
(After all transducers
are intersected and composed)
(~1M States, 1.6M Transitions)

22K Nouns
4K Verbs
2K Adjective
100K Proper Nouns

Finite State Transducers Employed



(e - v)ev+Noun+A3sg(i)+P3sg(n - "d e)+Loc
(e - v)ev+Noun+A3sg(i n)+P2sg(- "d e)+Loc

Morphology+Pronuncation

Resulting FST has 10M states 16M transitions

Total of 750 xfst regular expressions + 100K root words (mostly proper names) over about 50 files

Stress Computation Transducer

Syllabification Transducer

Exceptional Phonology Transducer

SAMPA Mapping Transducer

•ev+i+nde
•ev+in+de

Surface Morpheme Sequence

Filters

•ev+sH+ndA
•ev+Hn+DA

Lexical Morpheme Sequence

•ev+Noun+A3sg+P3sg+Loc
•ev+Noun+A3sg+P2sg+Loc

Feature Form

Modified Inverse Two-Level Rule Transducer

Filters

Filters

(Duplicating) Lexicon and Morphotactic Constraints Transducer

Two-Level Rule Transducer

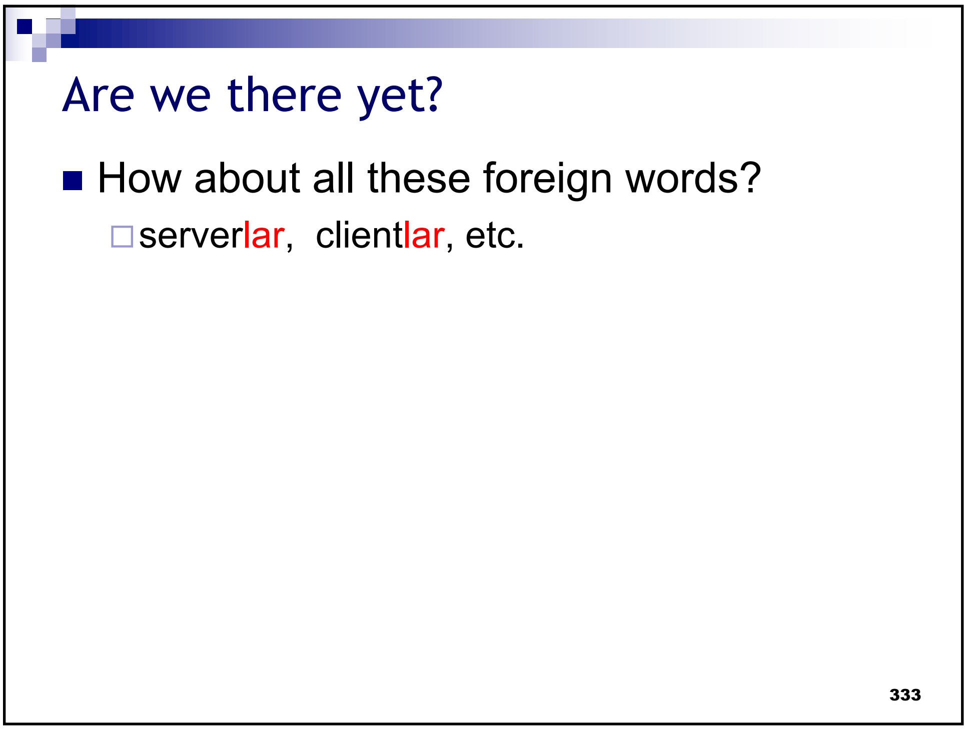
Surface form evinde

331

Pronunciation Generation

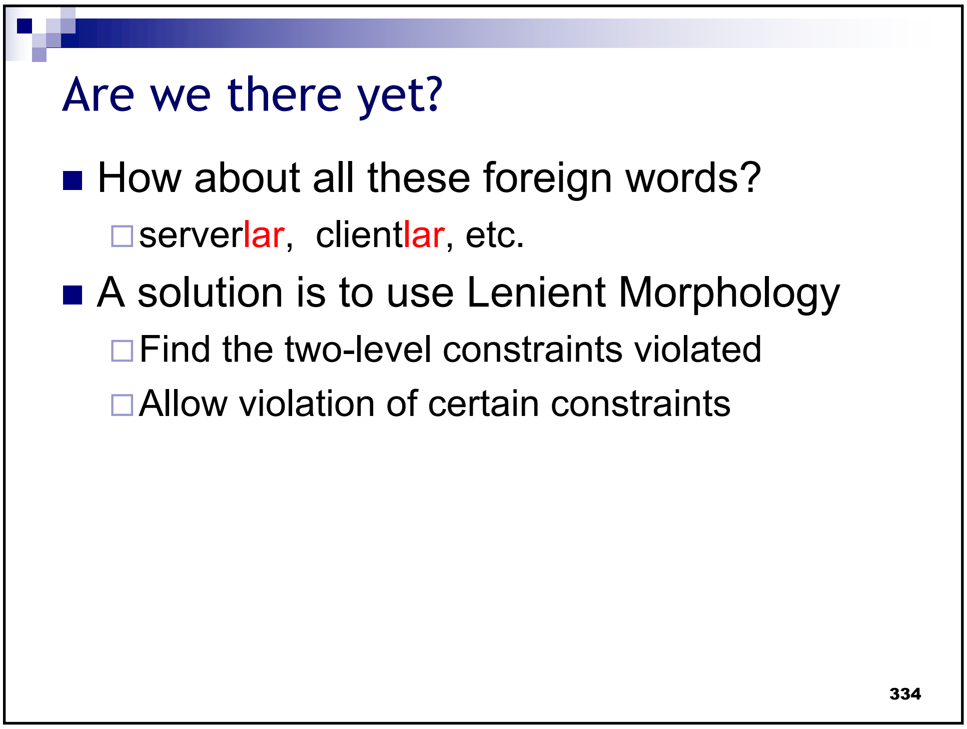
- gelebilecekken
  - □ (gj e - l)gel+Verb+Pos(e - b i - l) ^DB+Verb +Able(e - "dZ e c) +Fut(- c e n)^DB+Adverb+While
- okuma
  - □ (o - k)ok+Noun+A3sg(u - "m) +P1sg(a)+Dat
  - □ (o - "k u) oku+Verb(- m a) +Neg+Imp+A2sg
  - □ (o - k u) oku+Verb+Pos(- "m a) ^DB+Noun+Inf2+A3sg+Pnon+Nom

332



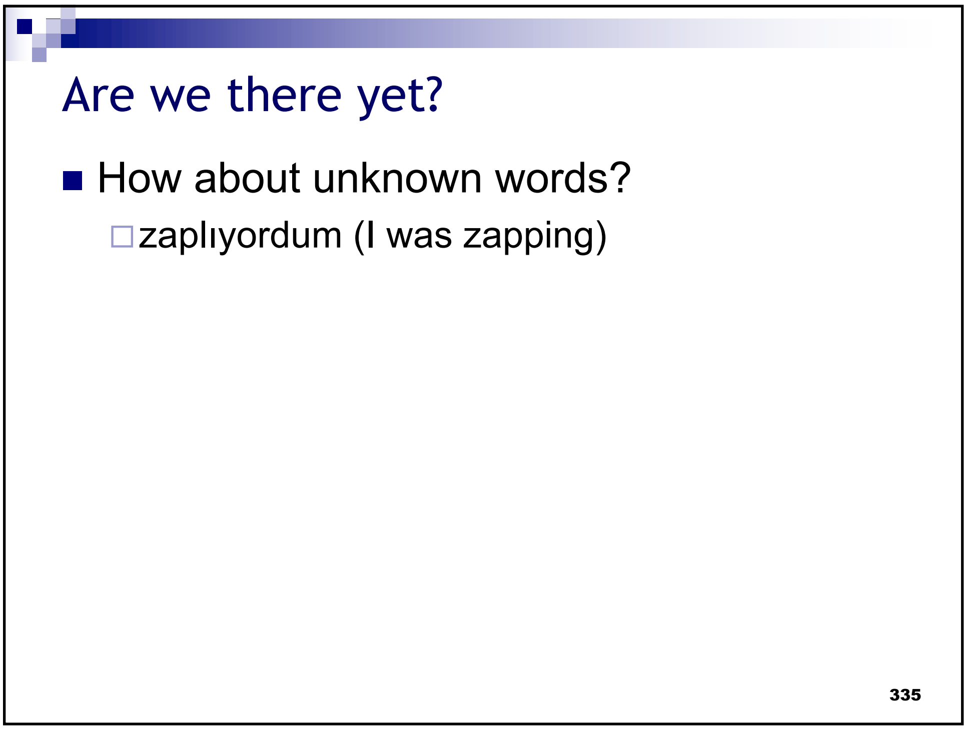
Are we there yet?

- How about all these foreign words?
  - □ serverlar, clientlar, etc.



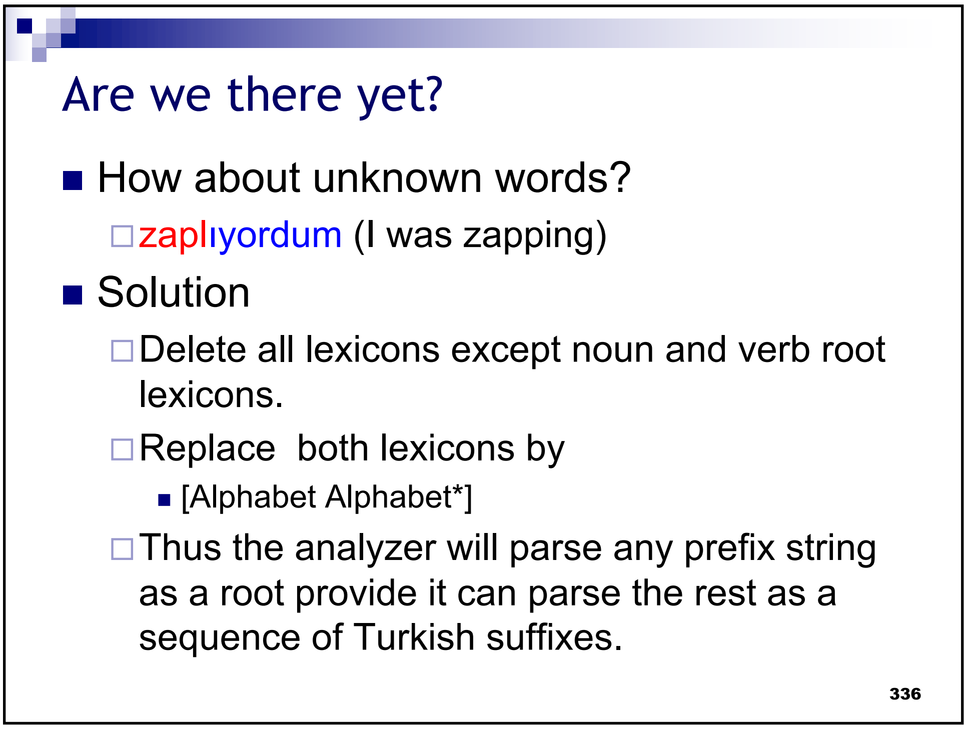
Are we there yet?

- How about all these foreign words?
  - □ serverlar, clientlar, etc.
- A solution is to use Lenient Morphology
  - □ Find the two-level constraints violated
  - □ Allow violation of certain constraints



Are we there yet?

- How about unknown words?
  - □ zaplıyordum (I was zapping)



Are we there yet?

- How about unknown words?
  - □ zaplıyordum (I was zapping)
- Solution
  - □ Delete all lexicons except noun and verb root lexicons.
  - □ Replace both lexicons by
    - [Alphabet Alphabet*]
  - □ Thus the analyzer will parse any prefix string as a root provide it can parse the rest as a sequence of Turkish suffixes.

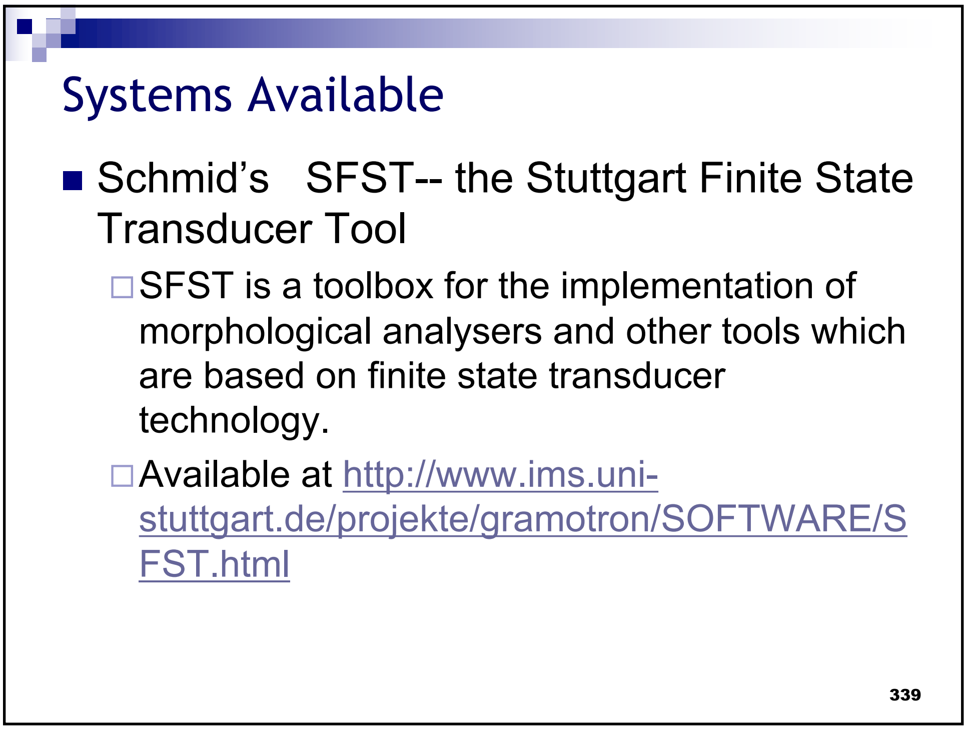
Are we there yet?

- How about unknown words?
  - □ zaplıyordum (I was zapping)

- Solution
  - □ zaplıyordum
    - `zapla+Hyor+DHm (zapla+Verb+Pos+Pres1+A1sg)`
    - `zapl +Hyor+DHm (zapl+Verb+Pos+Pres1+A1sg)`

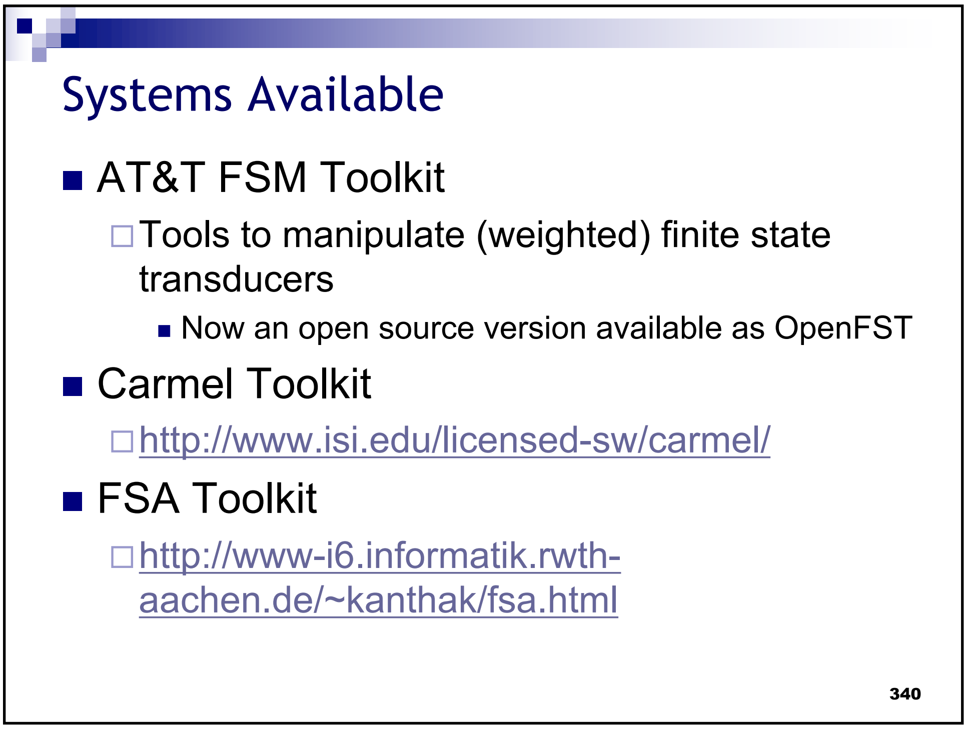
Systems Available

- Xerox Finite State Suite (lexc, twolc,xfst)
  - □ Commercial (Education/research license available)
  - □ Lexicon and rule compilers available
  - □ Full power of finite state calculus (beyond two-level morphology)
  - □ Very fast (thousands of words/sec)
- Beesley and Karttunen, Finite State Morphology, CSLI Publications, 2004 (www.fsmbook.com) comes with (a version of) this software
  - □ New versions of the software now available on the web via a click-through license (with bonus C++ and Pyhton API)



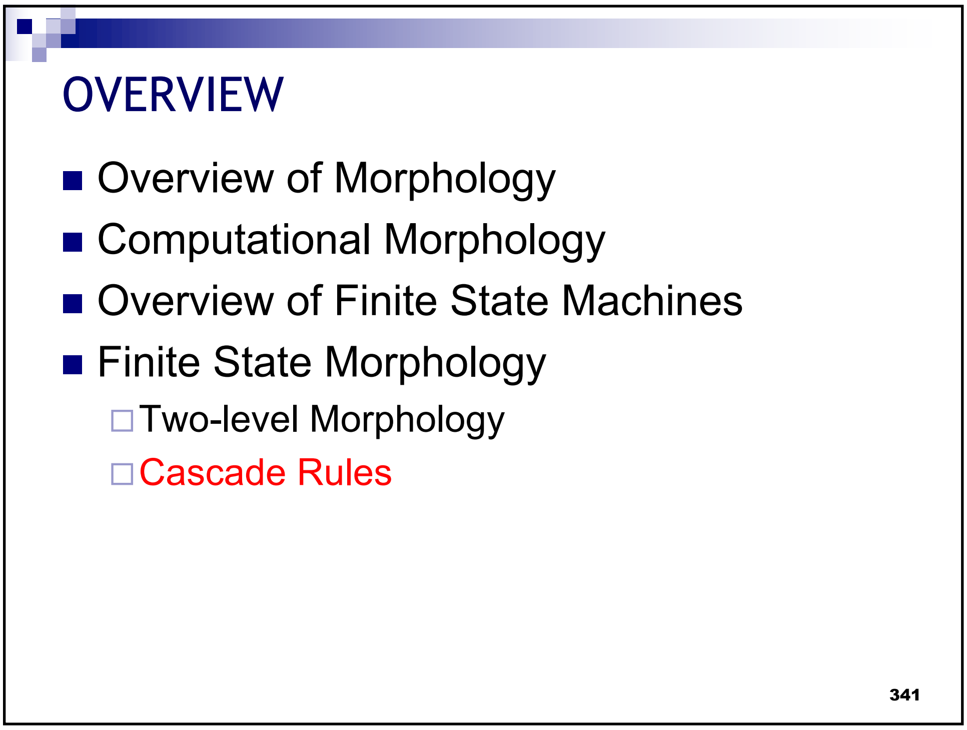
Systems Available

- Schmid's SFST-- the Stuttgart Finite State Transducer Tool
  - □ SFST is a toolbox for the implementation of morphological analysers and other tools which are based on finite state transducer technology.
  - □ Available at http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/SFST.html



Systems Available

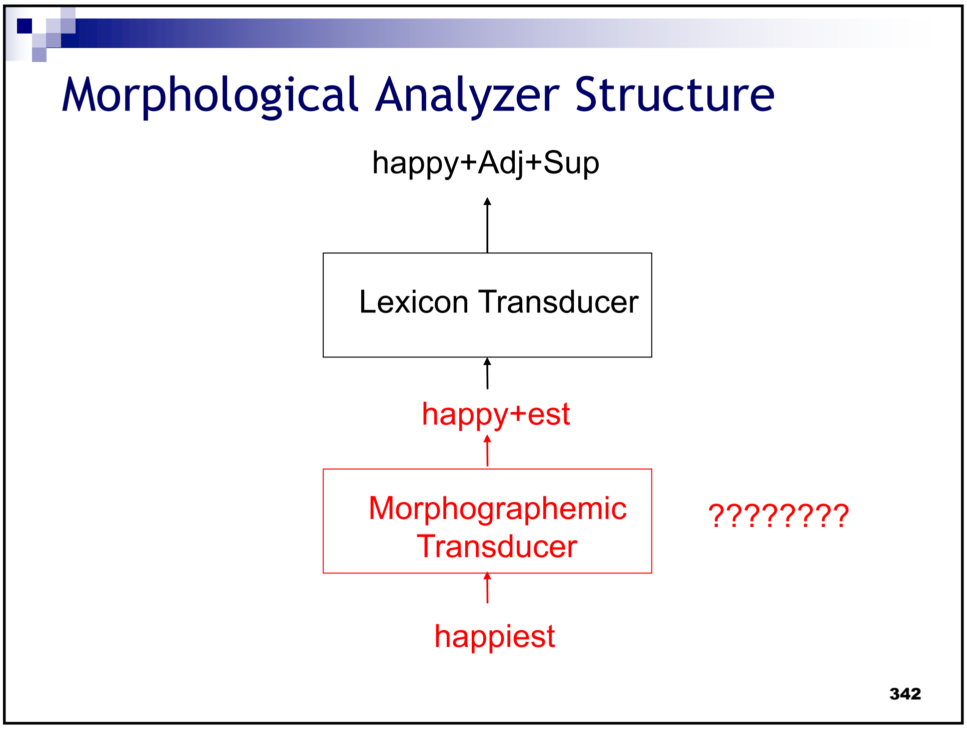
- AT&T FSM Toolkit
  - □ Tools to manipulate (weighted) finite state transducers
    - Now an open source version available as OpenFST
- Carmel Toolkit
  - □ http://www.isi.edu/licensed-sw/carmel/
- FSA Toolkit
  - □ http://www-i6.informatik.rwth-aachen.de/~kanthak/fsa.html



OVERVIEW

- Overview of Morphology
- Computational Morphology
- Overview of Finite State Machines
- Finite State Morphology
  - □ Two-level Morphology
  - □ Cascade Rules

341



Morphological Analyzer Structure

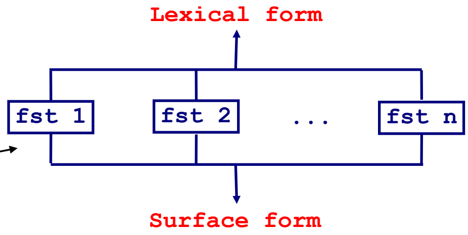
342

The Parallel Constraints Approach

- Two-level Morphology
  - □ Koskenniemi ’83, Karttunen et al. 1987, Karttunen et al. 1992

Describe constraints

Set of parallel of two-level rules compiled into finite-state automata interpreted as transducers



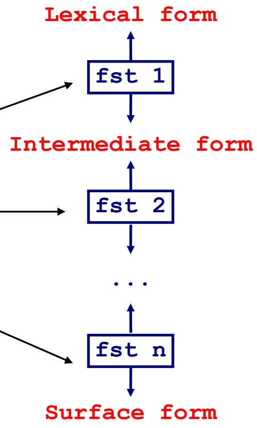
Slide courtesy of Lauri Karttunen

343

Sequential Transduction Approach

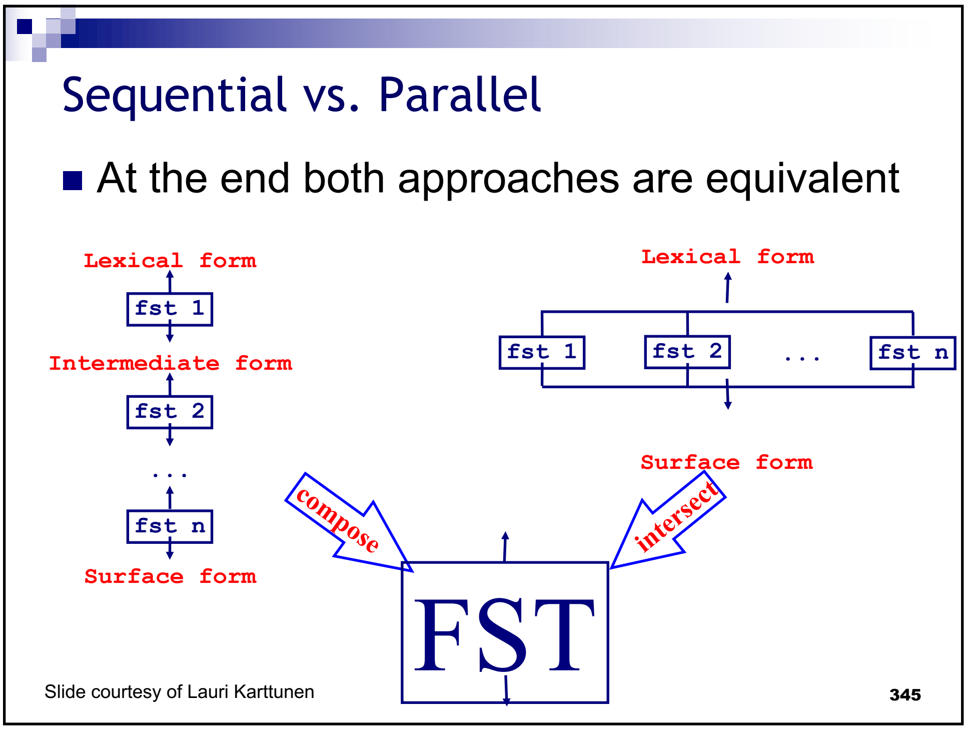
- Cascaded ordered rewrite rules
  - □ Johnson ’71 Kaplan & Kay ’81 based on Chomsky & Halle ‘64

Ordered cascade of rewrite rules compiled into finite-state transducers



Slide courtesy of Lauri Karttunen

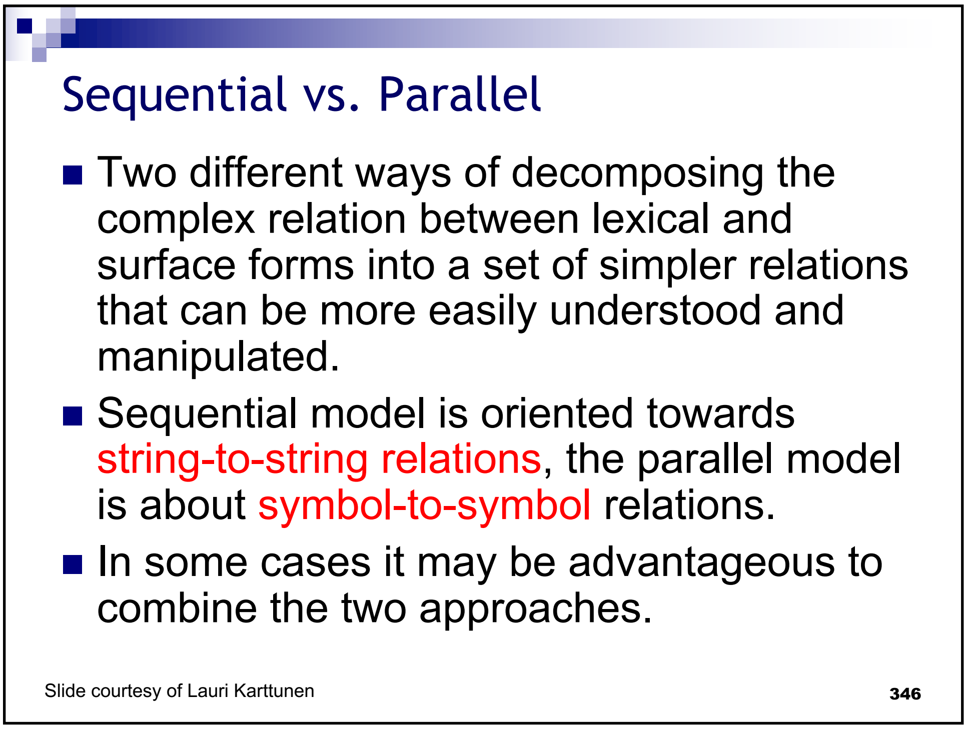
344



Sequential vs. Parallel

- At the end both approaches are equivalent

Slide courtesy of Lauri Karttunen

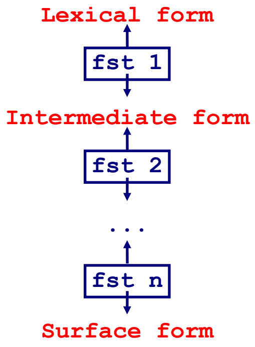


Sequential vs. Parallel

- Two different ways of decomposing the complex relation between lexical and surface forms into a set of simpler relations that can be more easily understood and manipulated.
- Sequential model is oriented towards string-to-string relations, the parallel model is about symbol-to-symbol relations.
- In some cases it may be advantageous to combine the two approaches.

Slide courtesy of Lauri Karttunen

Cascaded Ordered Rewrite Rules



- In this architecture, one transforms the lexical form to the surface form (or vice versa) through a series of transformations.
- In a sense, it is a procedural model.
  - □ What do I do to a lexical string to convert to a surface string?

The kaNpat Example (again)

- A lexical N before a lexical p is realized on the surface as an m.
- A p after an m is realized as m on the surface.

The kaNpat Example

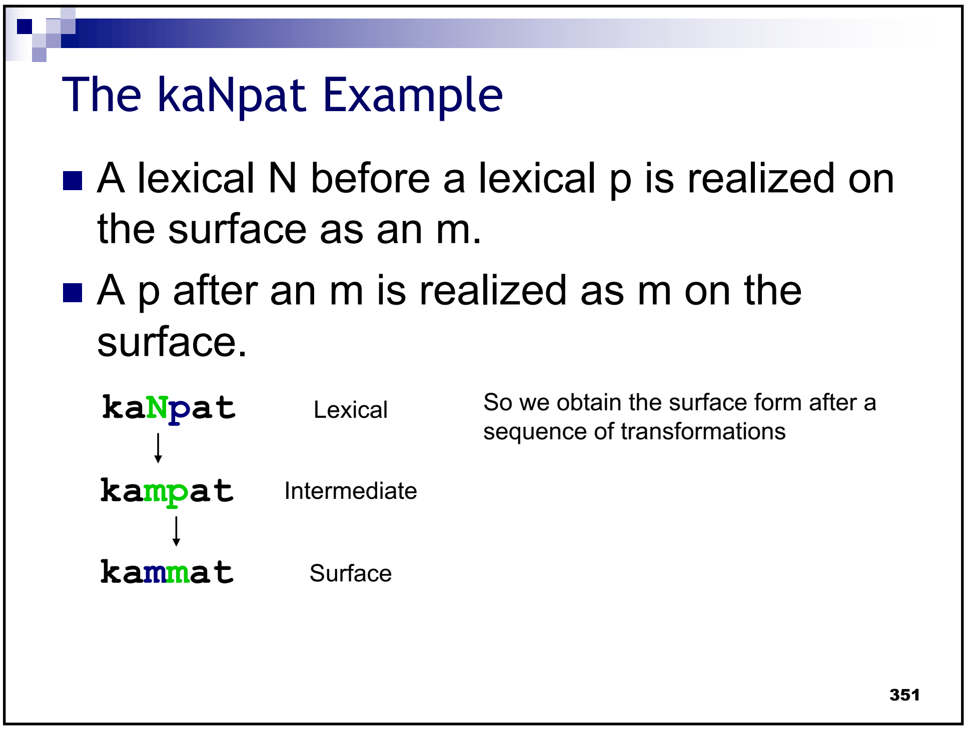
- A lexical N before a lexical p is realized on the surface as an m.
- A p after an m is realized as m on the surface.

```
...N p...
   ↓
...m p...
```

The kaNpat Example

- A lexical N before a lexical p is realized on the surface as an m.
- A p after an m is realized as m on the surface.

```
...N p...
   ↓
...m p...
     ↓
...m m...
```



The kaNpat Example

- A lexical N before a lexical p is realized on the surface as an m.
- A p after an m is realized as m on the surface.

kaNpat Lexical

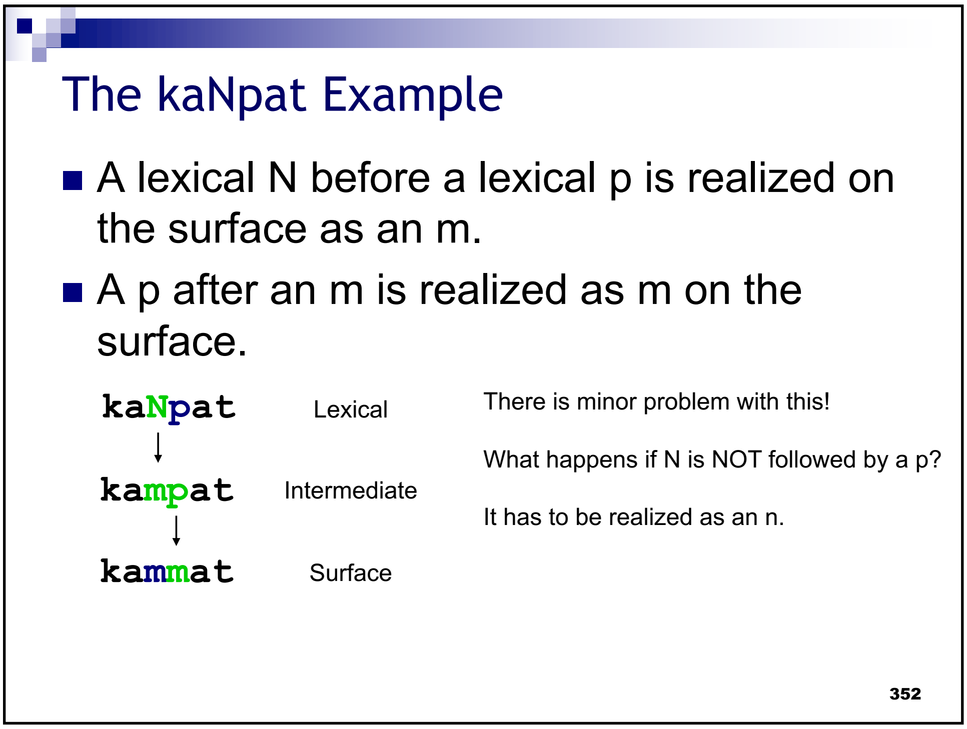
↓

kampat Intermediate

↓

kammat Surface

So we obtain the surface form after a sequence of transformations



The kaNpat Example

- A lexical N before a lexical p is realized on the surface as an m.
- A p after an m is realized as m on the surface.

kaNpat Lexical

↓

kampat Intermediate

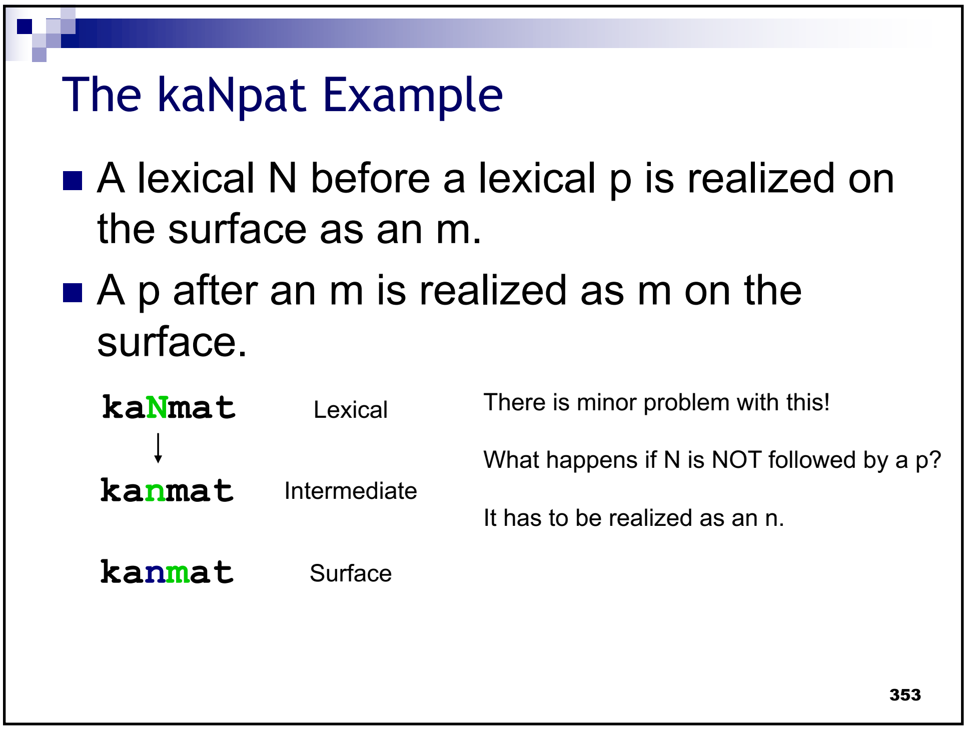
↓

kammat Surface

There is minor problem with this!

What happens if N is NOT followed by a p?

It has to be realized as an n.



The kaNpat Example

- A lexical N before a lexical p is realized on the surface as an m.
- A p after an m is realized as m on the surface.

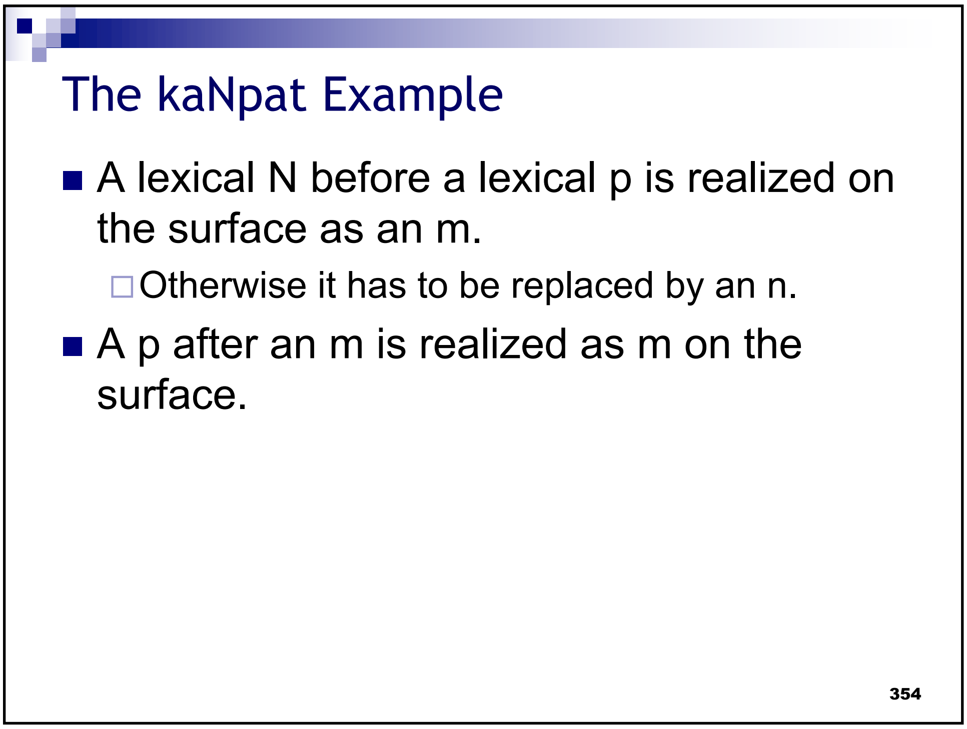
kaNmat	Lexical
↓	
kanmat	Intermediate
kanmat	Surface

There is minor problem with this!

What happens if N is NOT followed by a p?

It has to be realized as an n.

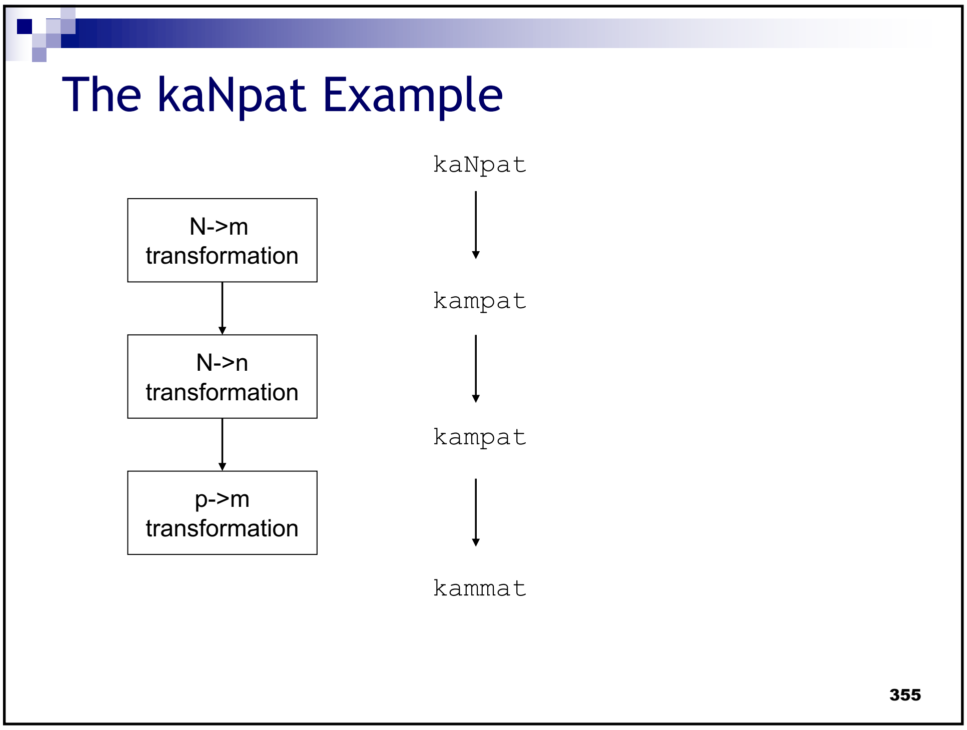
353



The kaNpat Example

- A lexical N before a lexical p is realized on the surface as an m.
  - □ Otherwise it has to be replaced by an n.
- A p after an m is realized as m on the surface.

354



The kaNpat Example

N->m transformation

N->n transformation

p->m transformation

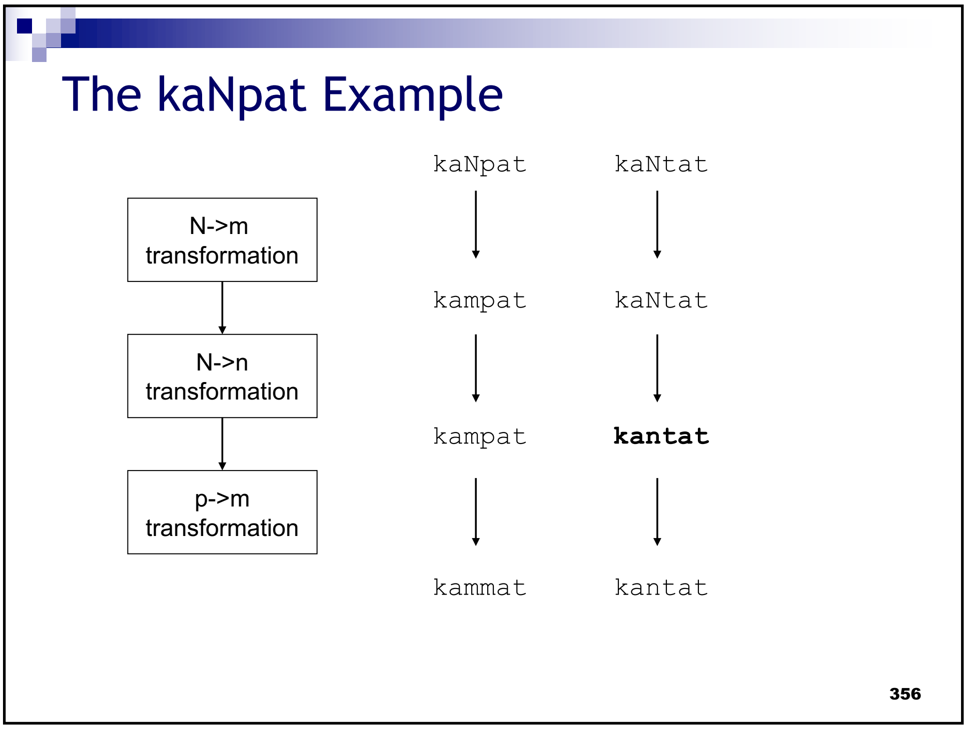
kaNpat

kampat

kampat

kammat

355



The kaNpat Example

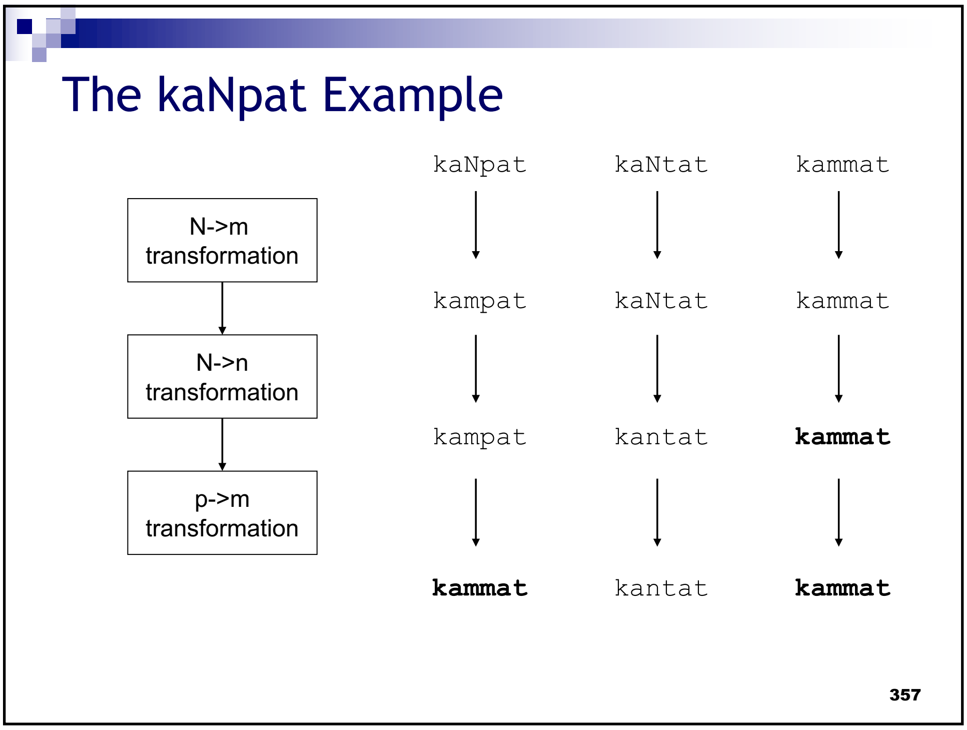
N->m transformation

N->n transformation

p->m transformation

kaNpat	kaNtat
kampat	kaNtat
kampat	**kantat**
kammat	kantat

356



The kaNpat Example

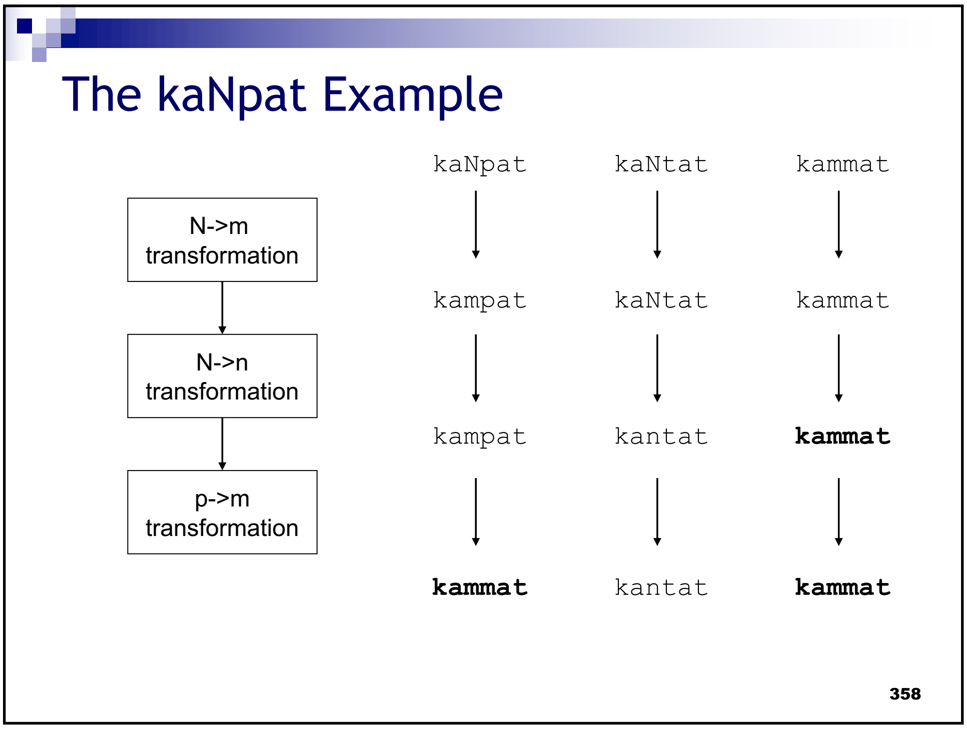
N->m transformation

N->n transformation

p->m transformation

kaNpat	kaNtat	kammat
kampat	kaNtat	kammat
kampat	kantat	**kammat**
kammat	kantat	**kammat**

357



The kaNpat Example

N->m transformation

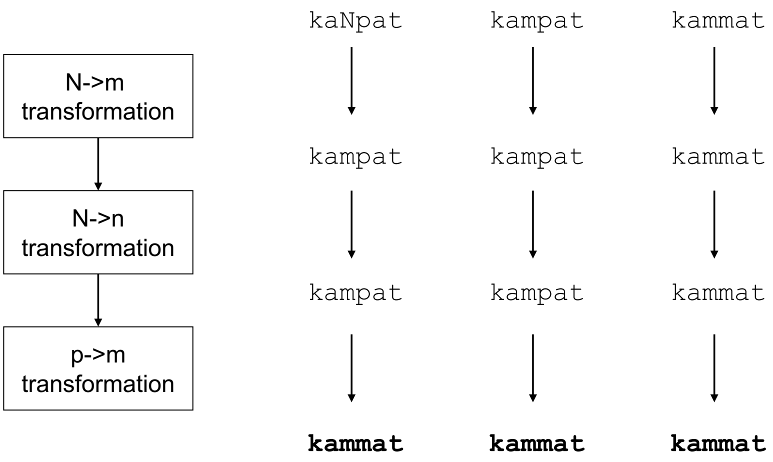
N->n transformation

p->m transformation

kaNpat	kaNtat	kammat
kampat	kaNtat	kammat
kampat	kantat	**kammat**
kammat	kantat	**kammat**

358

The kaNpat Example



N->m transformation

N->n transformation

p->m transformation

kaNpat kampat kammat

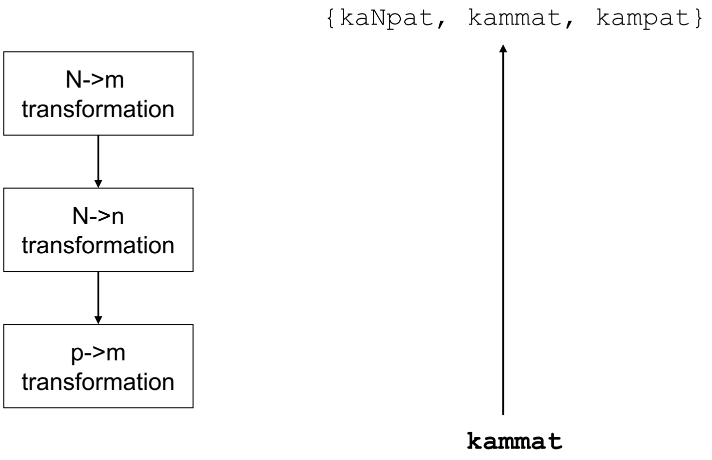
kampat kampat kammat

kampat kampat kammat

kammat **kammat** **kammat**

359

The kaNpat Example



N->m transformation

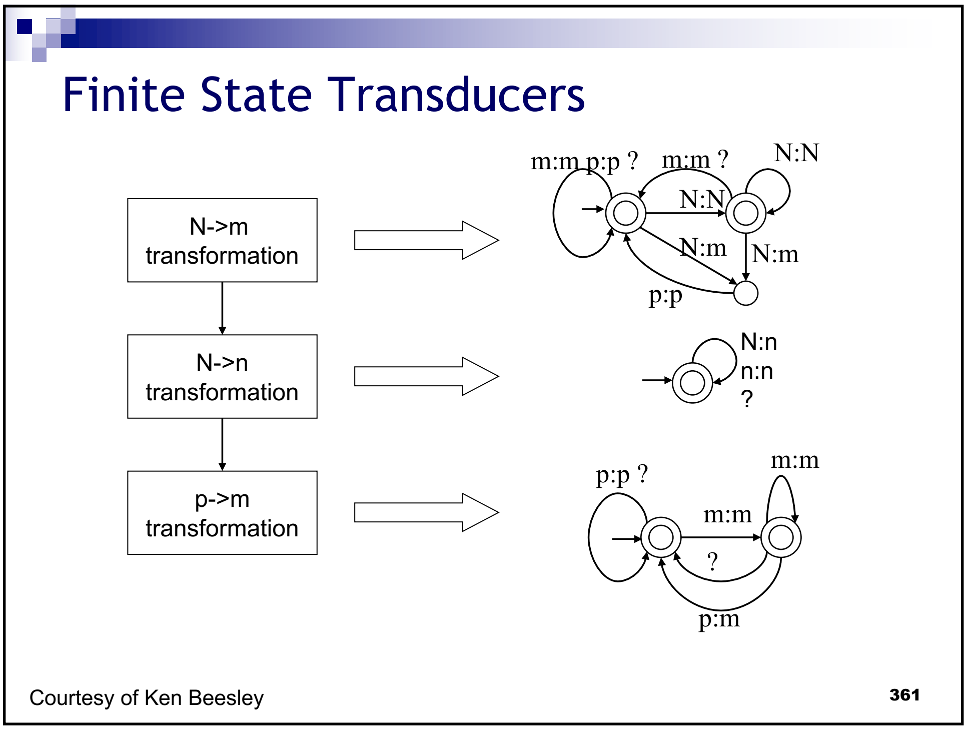
N->n transformation

p->m transformation

{kaNpat, kammat, kampat}

kammat

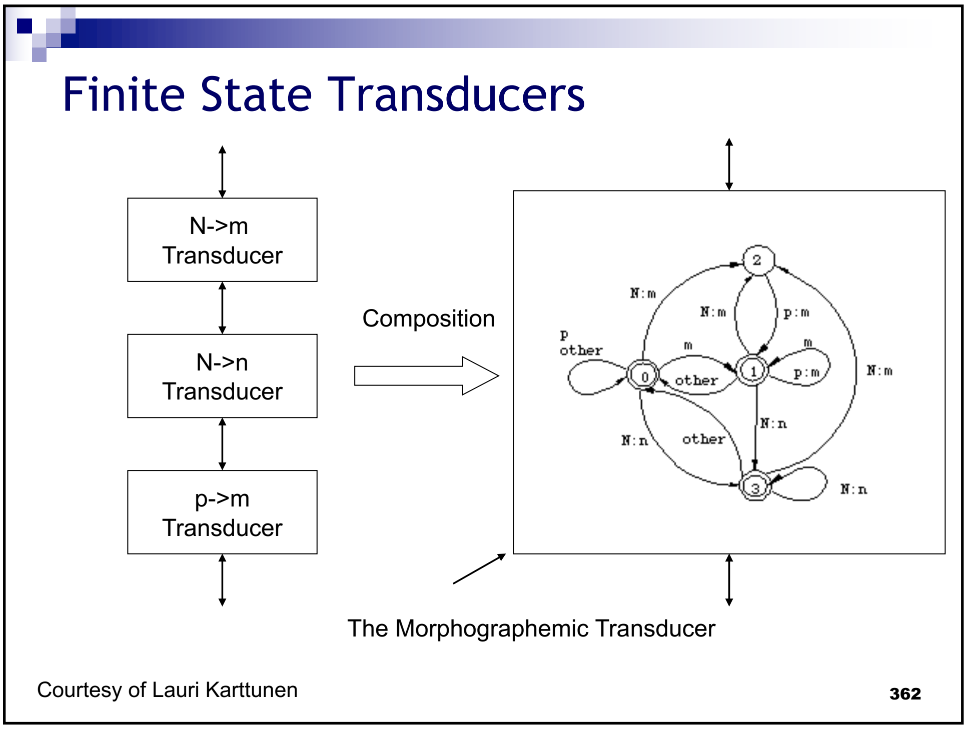
360



Finite State Transducers

Courtesy of Ken Beesley

361

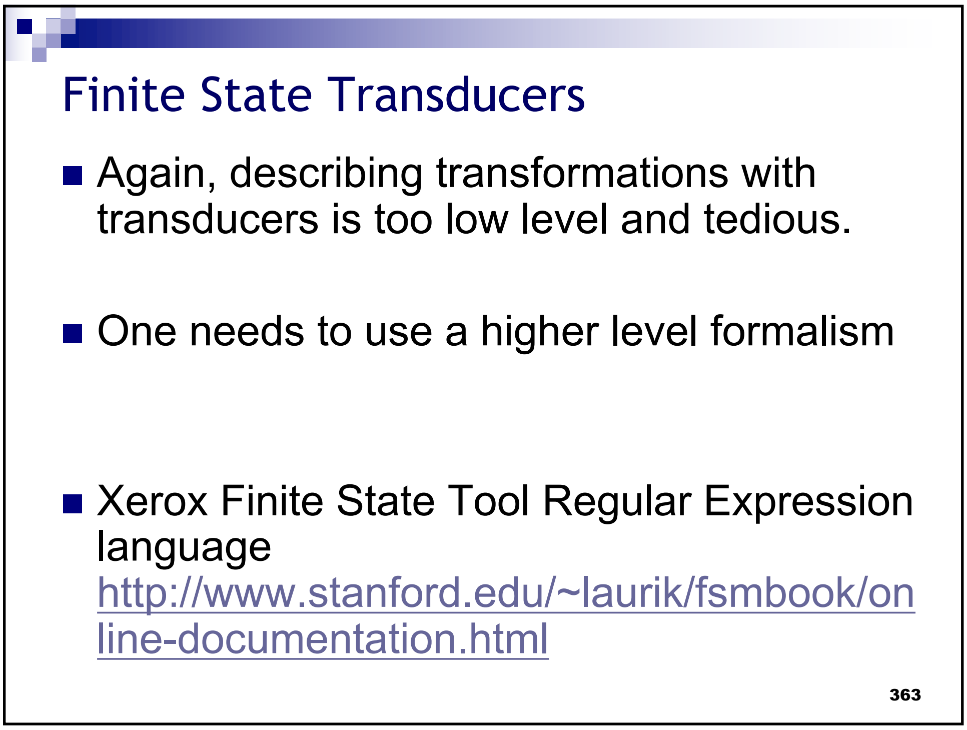


Finite State Transducers

The Morphographemic Transducer

Courtesy of Lauri Karttunen

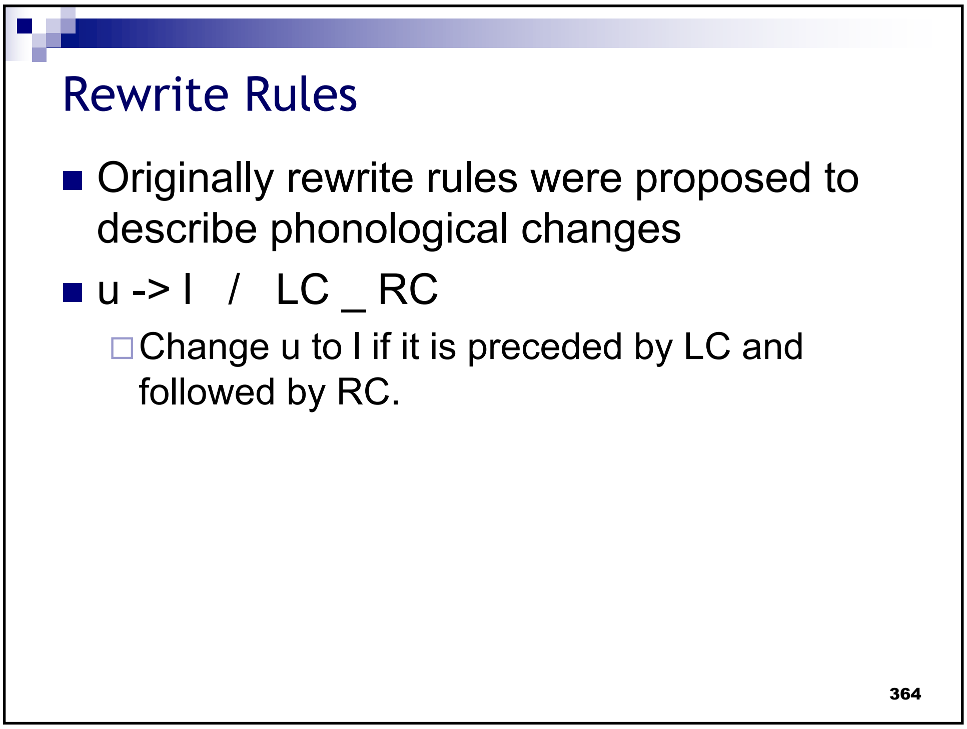
362



Finite State Transducers

- Again, describing transformations with transducers is too low level and tedious.
- One needs to use a higher level formalism
- Xerox Finite State Tool Regular Expression language http://www.stanford.edu/~laurik/fsmbook/online-documentation.html

363



Rewrite Rules

- Originally rewrite rules were proposed to describe phonological changes
- u -> l / LC _ RC
  - □ Change u to l if it is preceded by LC and followed by RC.

364

Rewrite Rules

- These rules of the sort u -> l / LC _ RC look like context sensitive grammar rules, so can in general describe much more complex languages.
- Johnson (1972) showed that such rewrite rules are only finite-state in power, if some constraints on how they apply to strings are imposed.

365

Replace Rules

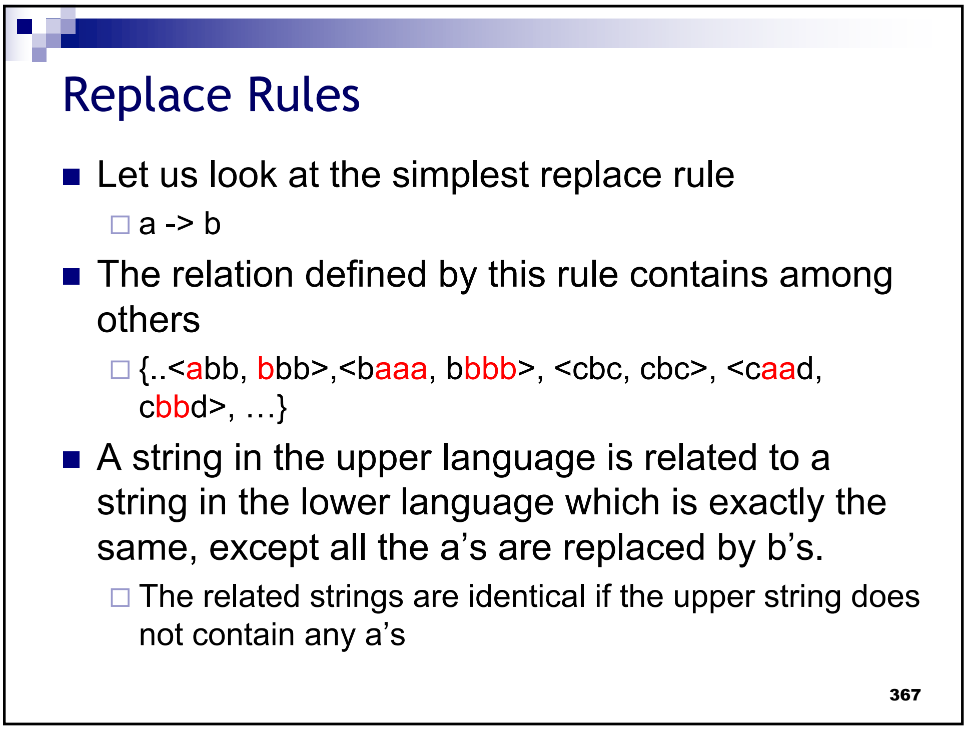
- Replace rules define regular relations between two regular languages

A -> B	LC _ RC
Replacement	Context

The relation that replaces **A** by **B** between **L** and **R** leaving everything else unchanged.

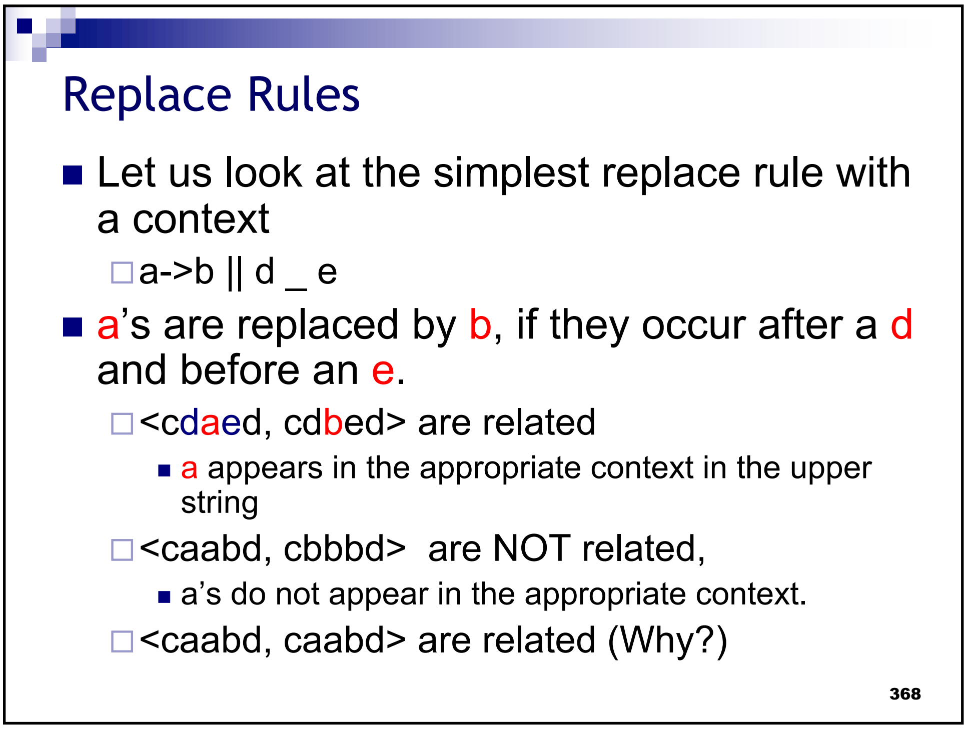
- In general A, B, LC and RC are regular expressions.

366



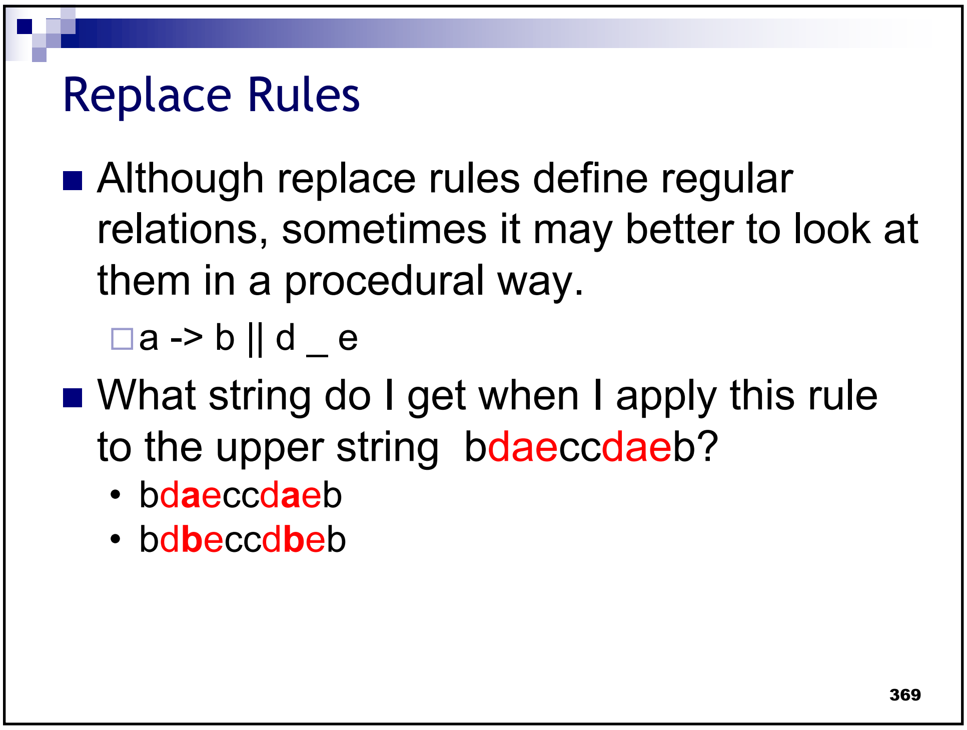
Replace Rules

- Let us look at the simplest replace rule
  - □ a -> b
- The relation defined by this rule contains among others
  - □ {..<abb, bbb>,<baaa, bbbb>, <cbc, cbc>, <caad, cbbd>, …}
- A string in the upper language is related to a string in the lower language which is exactly the same, except all the a’s are replaced by b’s.
  - □ The related strings are identical if the upper string does not contain any a’s



Replace Rules

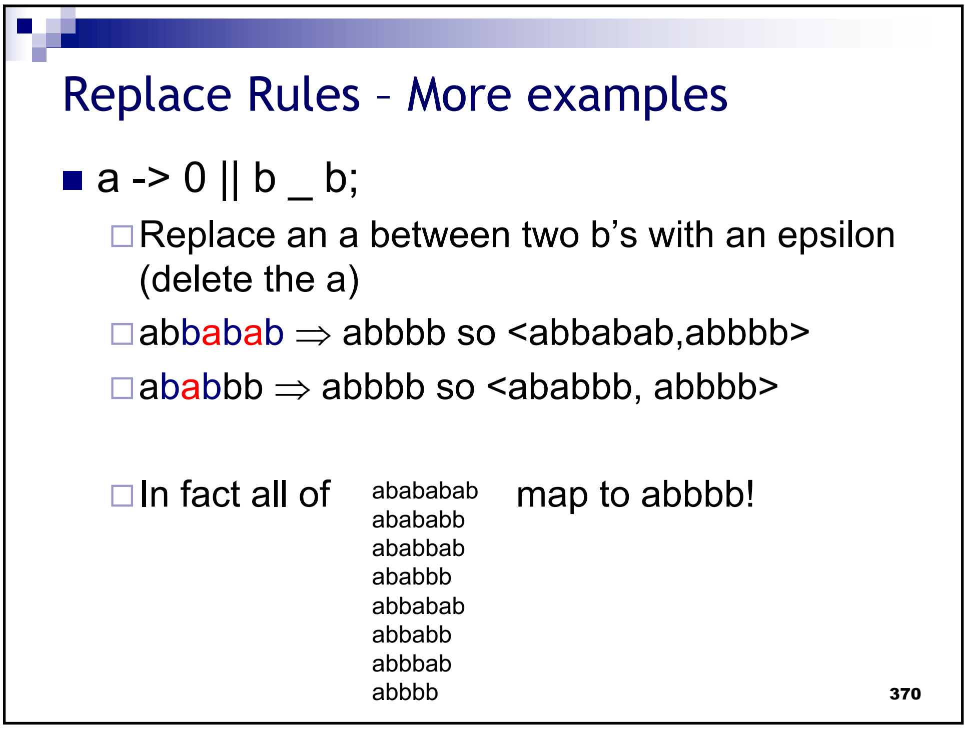
- Let us look at the simplest replace rule with a context
  - □ a->b || d _ e
- a’s are replaced by b, if they occur after a d and before an e.
  - □ <cdaed, cdbed> are related
    - a appears in the appropriate context in the upper string
  - □ <caabd, cbbbd> are NOT related,
    - a’s do not appear in the appropriate context.
  - □ <caabd, caabd> are related (Why?)



Replace Rules

- Although replace rules define regular relations, sometimes it may better to look at them in a procedural way.
  - □a -> b || d _ e
- What string do I get when I apply this rule to the upper string bdaeccdaeb?
  - bdaeccdaeb
  - bdbeccdbeb

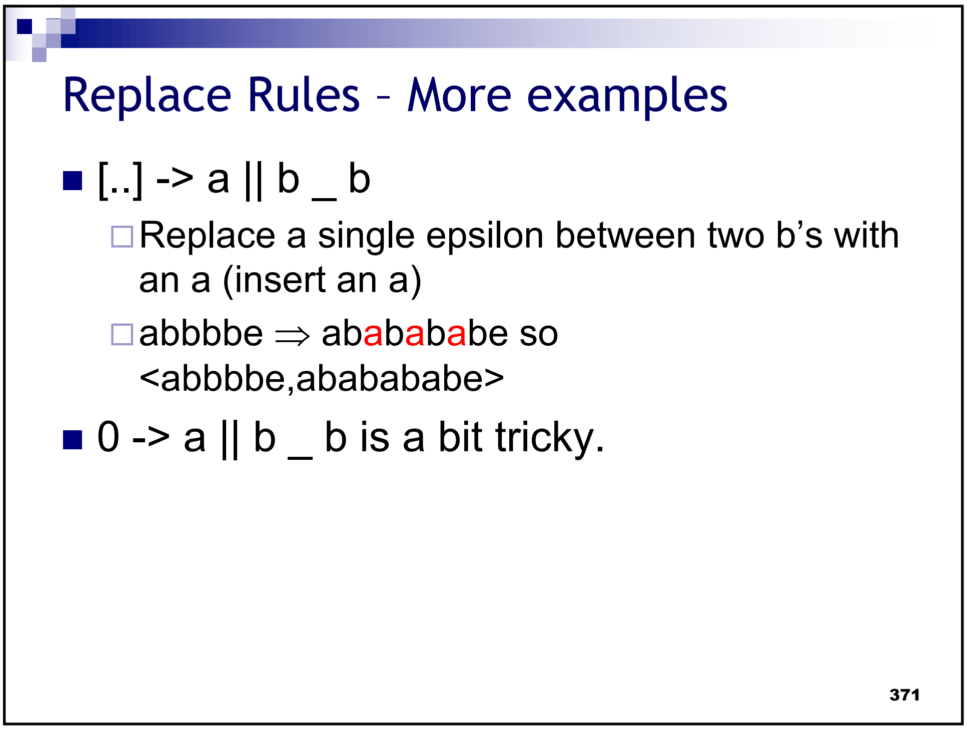
369



Replace Rules - More examples

- a -> 0 || b _ b;
  - □Replace an a between two b's with an epsilon (delete the a)
  - □abbabab ⇒ abbbb so <abbabab,abbbb>
  - □ababbb ⇒ abbbb so <ababbb, abbbb>
  - □In fact all of
    ababab
    abababb
    ababbab
    ababbb
    abbabab
    abbabb
    abbbab
    abbbb
    map to abbbb!

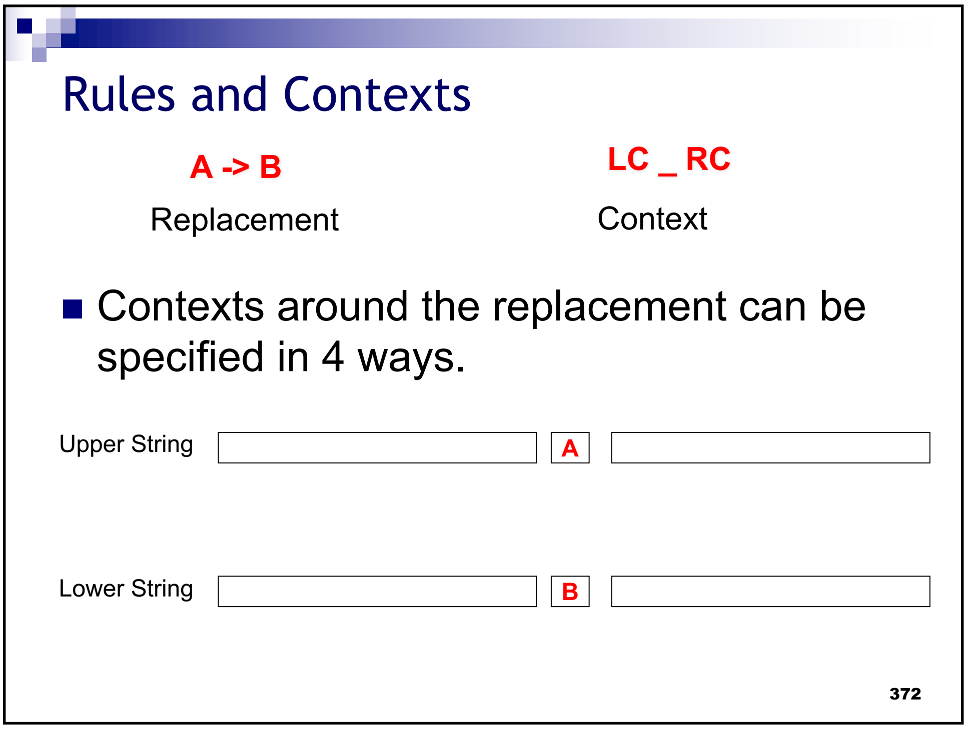
370



Replace Rules – More examples

- [..] -> a || b _ b
  - □ Replace a single epsilon between two b's with an a (insert an a)
  - □ abbbbe ⇒ ababababe so <abbbbe,ababababe>
- 0 -> a || b _ b is a bit tricky.

371



Rules and Contexts

A -> B Replacement

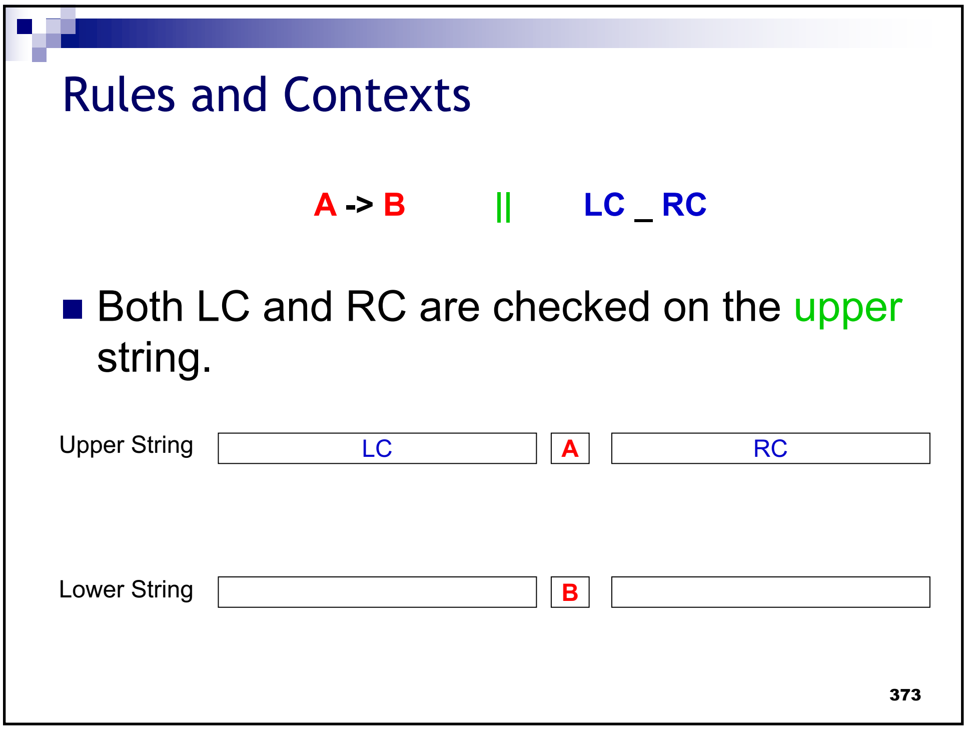
LC _ RC Context

- Contexts around the replacement can be specified in 4 ways.

Upper String A

Lower String B

372

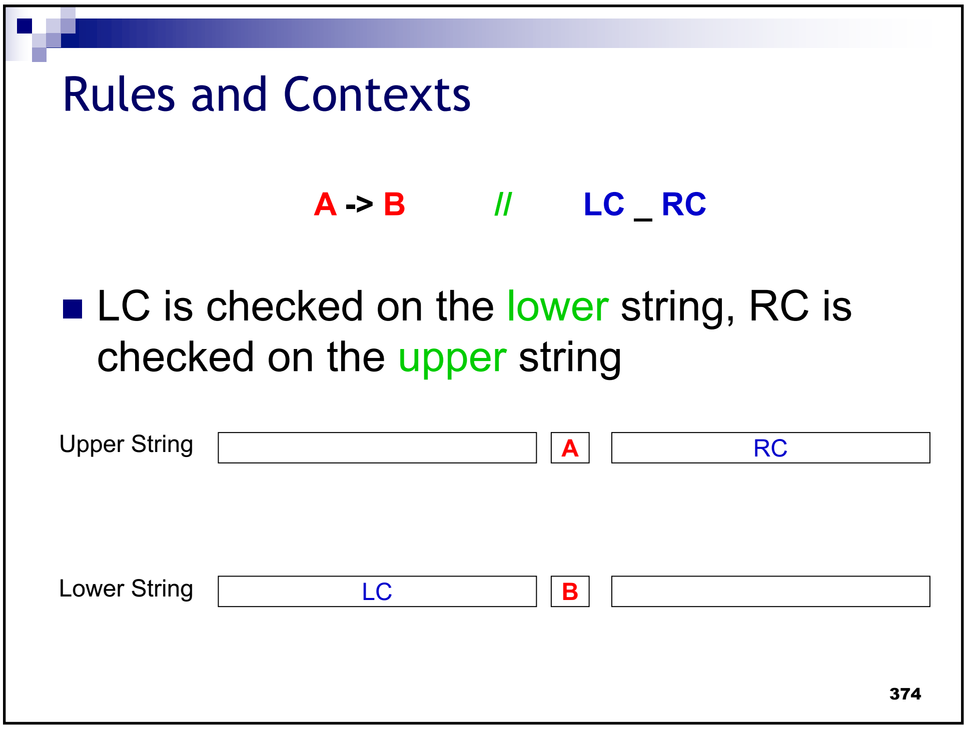


Rules and Contexts

A -> B || LC _ RC

- Both LC and RC are checked on the upper string.

Upper String	LC	A	RC
Lower String		B	

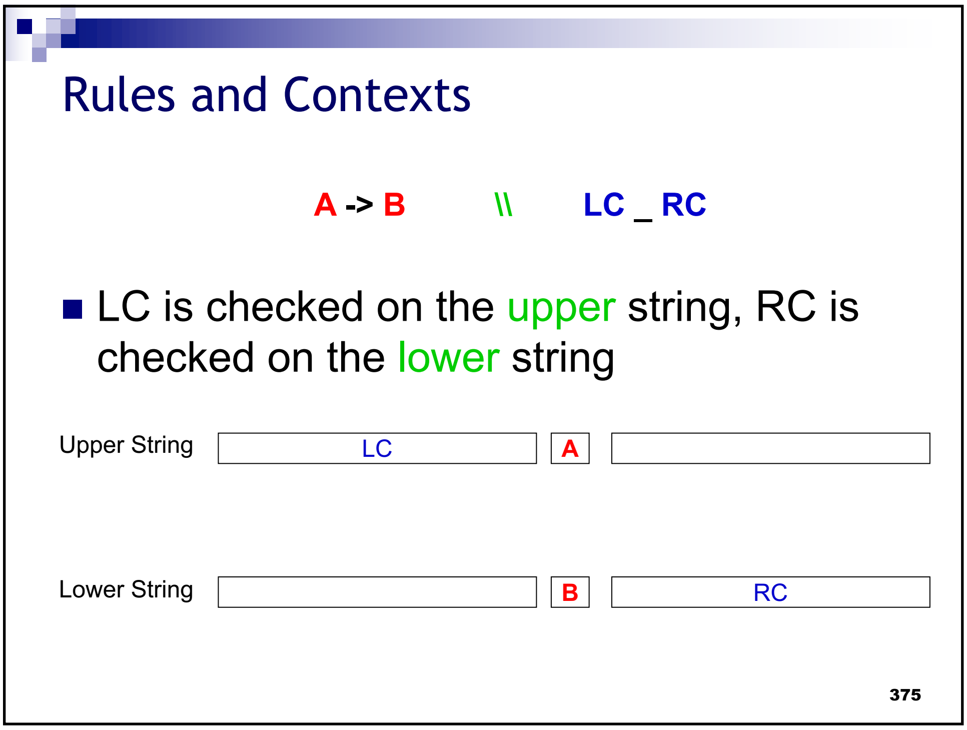


Rules and Contexts

A -> B // LC _ RC

- LC is checked on the lower string, RC is checked on the upper string

Upper String		A	RC
Lower String	LC	B	



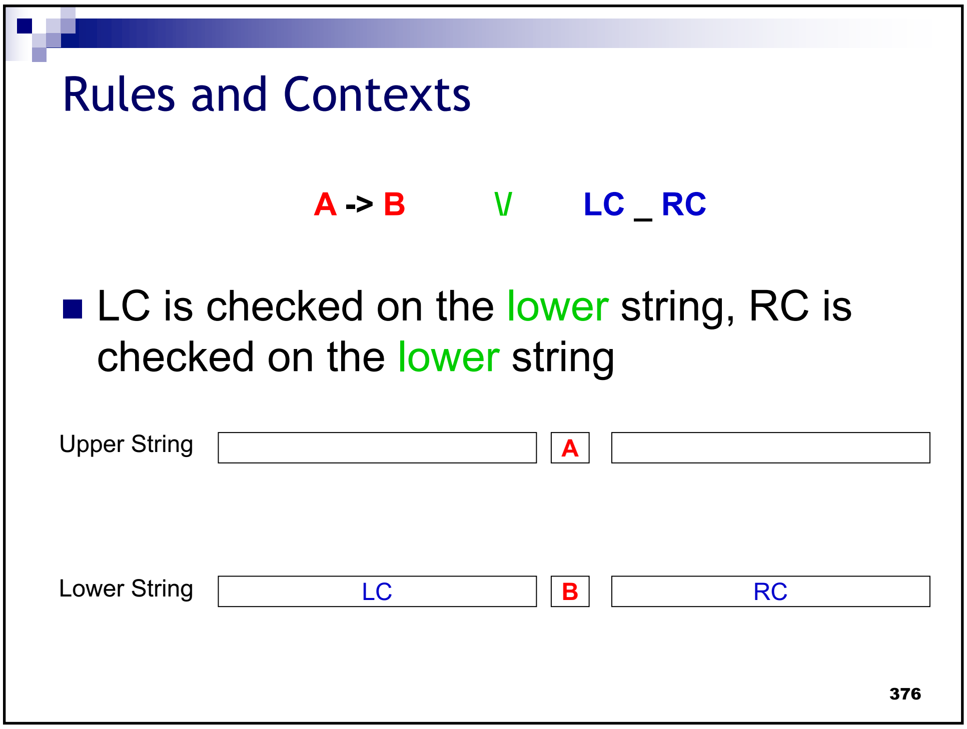
Rules and Contexts

A -> B \\ LC _ RC

- LC is checked on the upper string, RC is checked on the lower string

Upper String	LC	A	
Lower String		B	RC

375



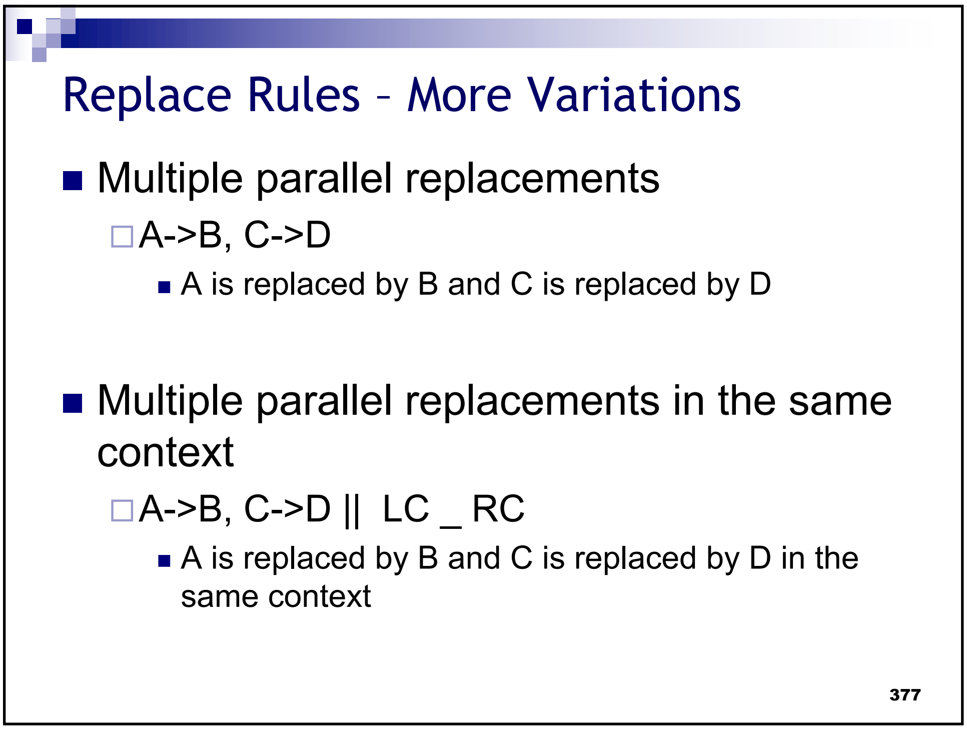
Rules and Contexts

A -> B \/ LC _ RC

- LC is checked on the lower string, RC is checked on the lower string

Upper String		A	
Lower String	LC	B	RC

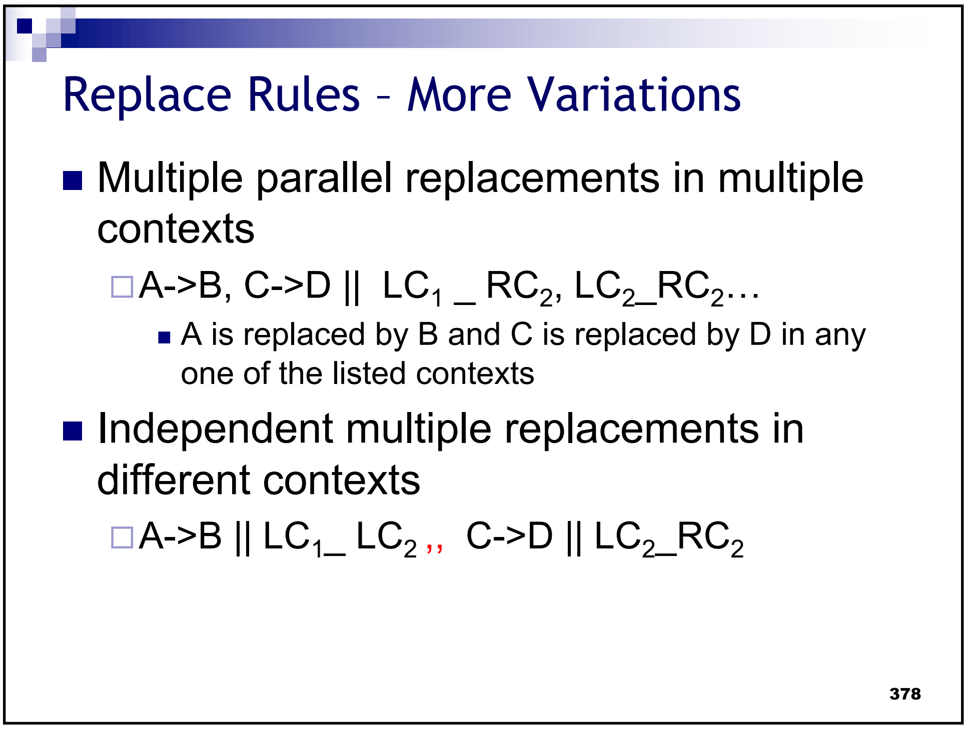
376



Replace Rules – More Variations

- Multiple parallel replacements
  - □A->B, C->D
    - A is replaced by B and C is replaced by D
- Multiple parallel replacements in the same context
  - □A->B, C->D || LC _ RC
    - A is replaced by B and C is replaced by D in the same context

377



Replace Rules – More Variations

- Multiple parallel replacements in multiple contexts
  - □A->B, C->D || LC_1 _ RC_2, LC_2_RC_2…
    - A is replaced by B and C is replaced by D in any one of the listed contexts
- Independent multiple replacements in different contexts
  - □A->B || LC_1_ LC_2 ,, C->D || LC_2_RC_2

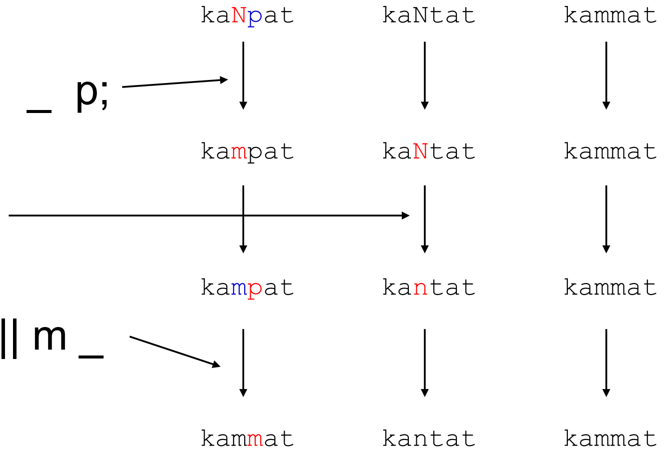
378

Replace Rules for KaNpat

- N->m || _ p;
  - □ replace N with m just before an upper p;
- N-> n;
  - □ Replace N with n otherwise
- p -> m || m _;
  - □ Replace p with an m just after an m (in the upper string

Replace Rules for KaNpat

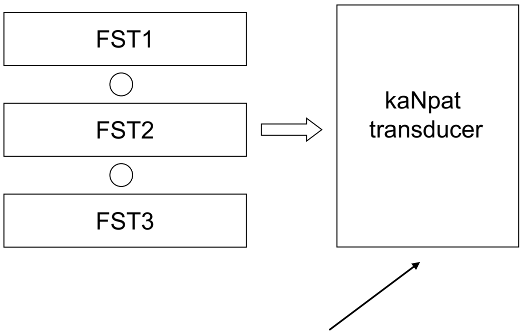
- N->m || _ p;
- N-> n;
- p -> m || m _



kaNpat	kaNtat	kammat
kampat	kaNtat	kammat
kampat	kantat	kammat
kammat	kantat	kammat

Replace Rules for KaNpat (combined)

N->m || _ p;
.o.
N-> n;
.o.
p -> m || m _



- Defines a single transducer that "composes" the three transducers.

381

A Cascade Rule Sequence for Turkish

- Remember the conventions
  - A = {a, e}, H= {ı, i, u, ü}
  - VBack = [a | ı | o | u] //the back vowels
  - VFront = [e | i | ö | ü] //the front vowels
  - Vowel = VBack | VFront
  - Consonant = […all consonants ..]

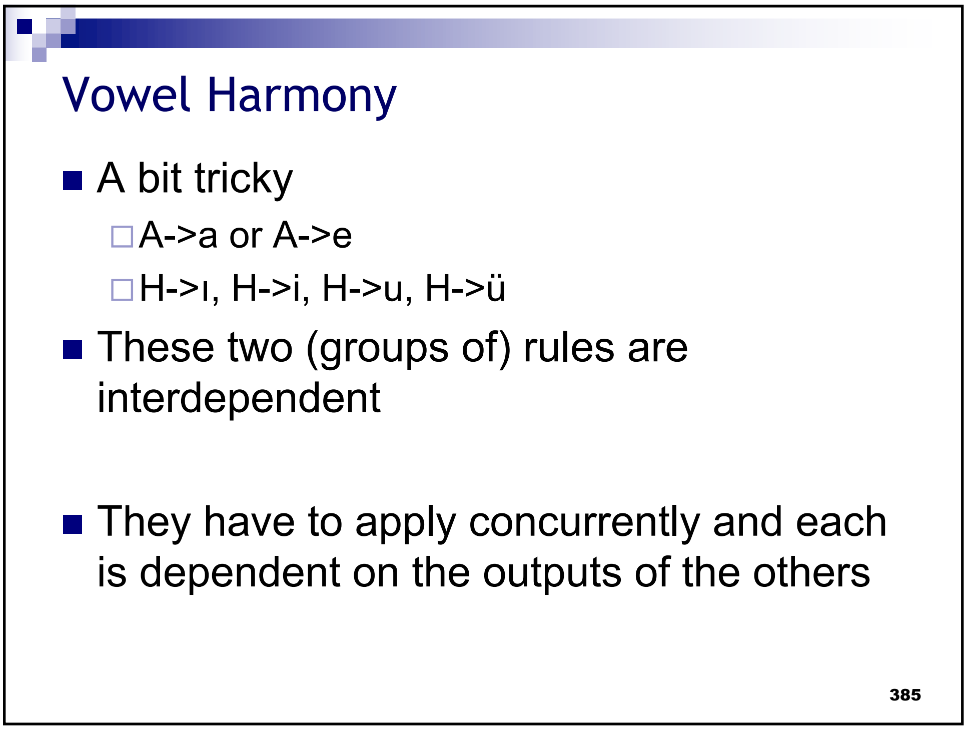
382

Stem Final Vowel Deletion

- A vowel ending a stem is deleted if it is followed by the morpheme +Hyor
  - □ata+Hyor (is assigning)
  - □at+Hyor
- Vowel ->0 || _ "+" H y o r

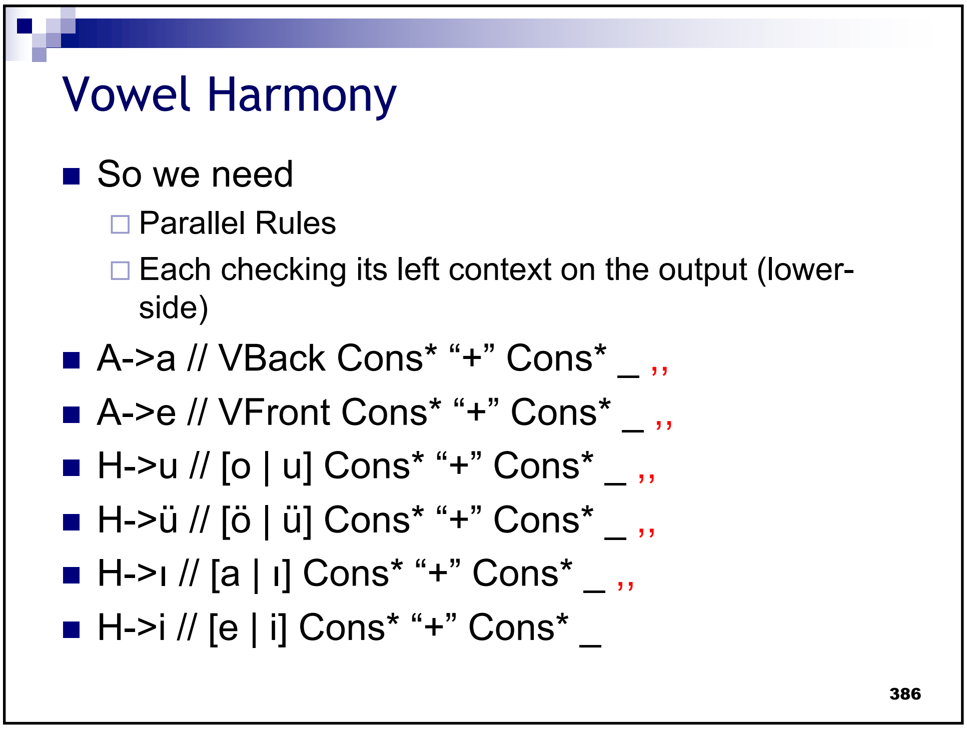
Morpheme Initial Vowel Deletion

- A high vowel starting a morpheme is deleted if it is attached to a segment ending in a vowel.
  - □masa+Hm (my table)
  - □masa+m
- H ->0 || Vowel "+" _ ;
- Note that the previous rule is a more special case rule



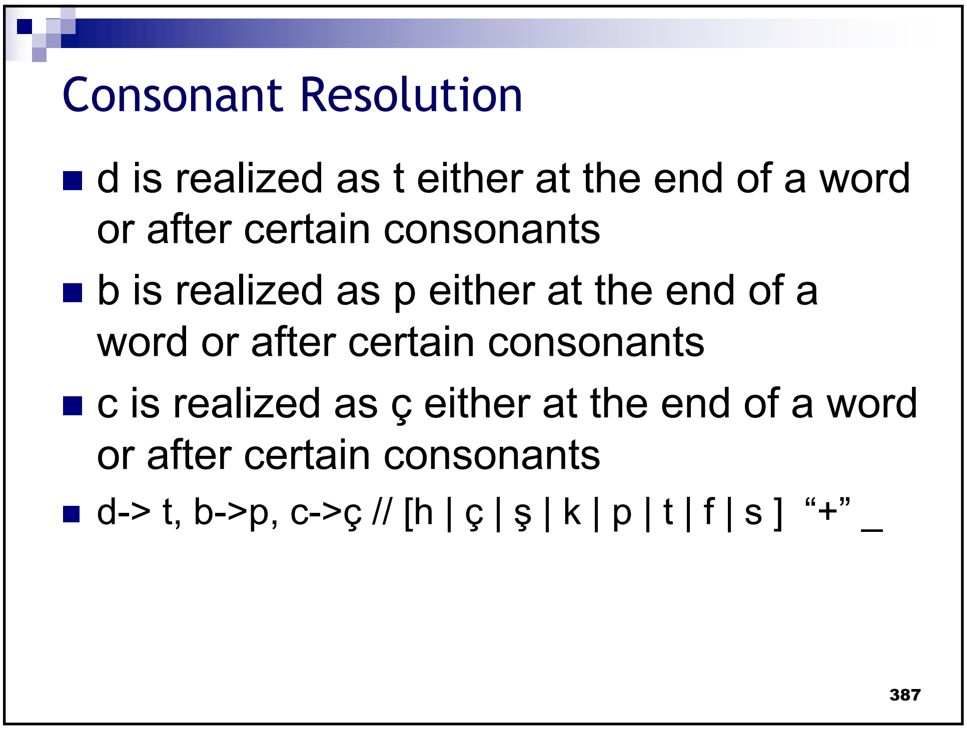
Vowel Harmony

- A bit tricky
  - □ A->a or A->e
  - □ H->ı, H->i, H->u, H->ü
- These two (groups of) rules are interdependent
- They have to apply concurrently and each is dependent on the outputs of the others



Vowel Harmony

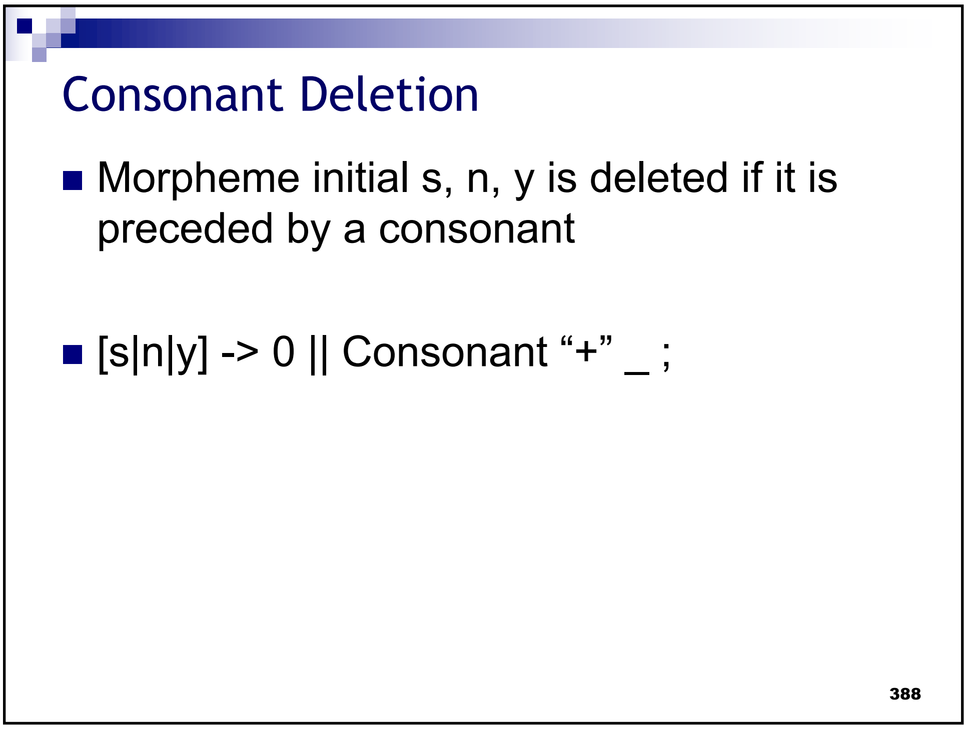
- So we need
  - □ Parallel Rules
  - □ Each checking its left context on the output (lower-side)
- A->a // VBack Cons* “+” Cons* _ ,,
- A->e // VFront Cons* “+” Cons* _ ,,
- H->u // [o | u] Cons* “+” Cons* _ ,,
- H->ü // [ö | ü] Cons* “+” Cons* _ ,,
- H->ı // [a | ı] Cons* “+” Cons* _ ,,
- H->i // [e | i] Cons* “+” Cons* _



Consonant Resolution

- d is realized as t either at the end of a word or after certain consonants
- b is realized as p either at the end of a word or after certain consonants
- c is realized as ç either at the end of a word or after certain consonants
- d-> t, b->p, c->ç // [h | ç | ş | k | p | t | f | s] “+” _

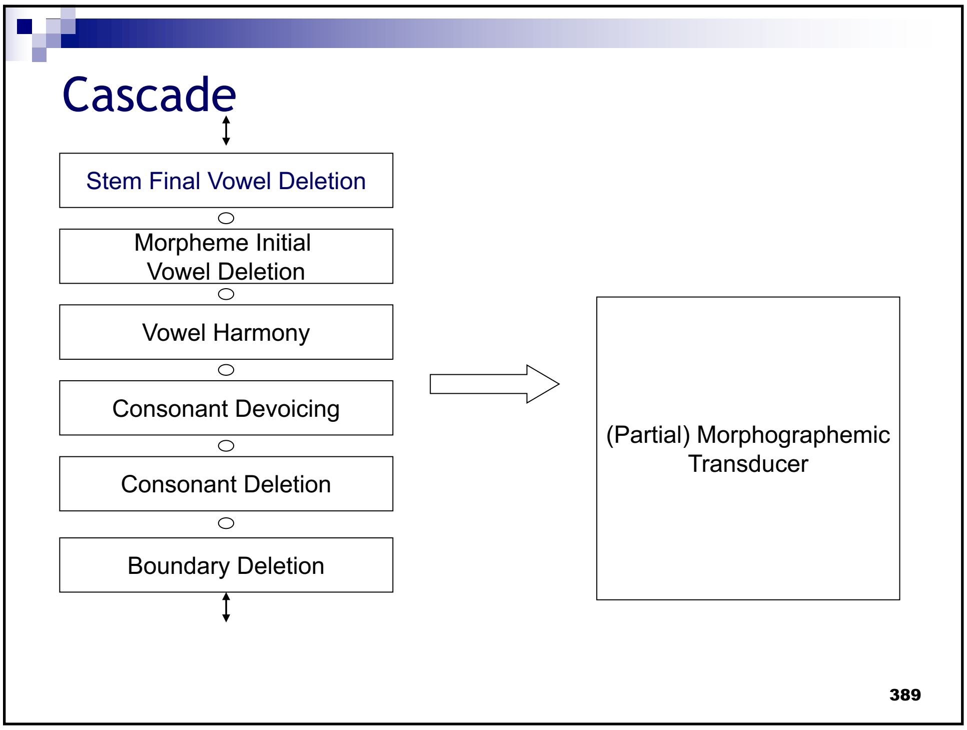
387



Consonant Deletion

- Morpheme initial s, n, y is deleted if it is preceded by a consonant
- [s|n|y] -> 0 || Consonant “+” _ ;

388



Cascade

↕

Stem Final Vowel Deletion

○

Morpheme Initial Vowel Deletion

○

Vowel Harmony

○

Consonant Devoicing

○

Consonant Deletion

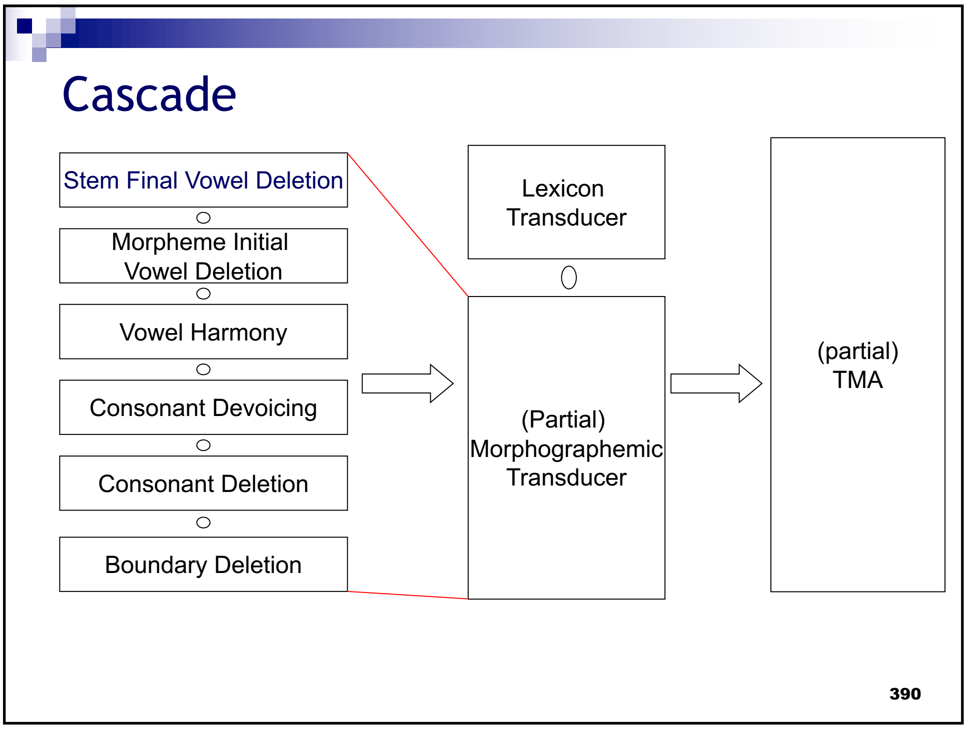
○

Boundary Deletion

↕

⇨

(Partial) Morphographemic Transducer



Cascade

Stem Final Vowel Deletion

○

Morpheme Initial Vowel Deletion

○

Vowel Harmony

○

Consonant Devoicing

○

Consonant Deletion

○

Boundary Deletion

⇨

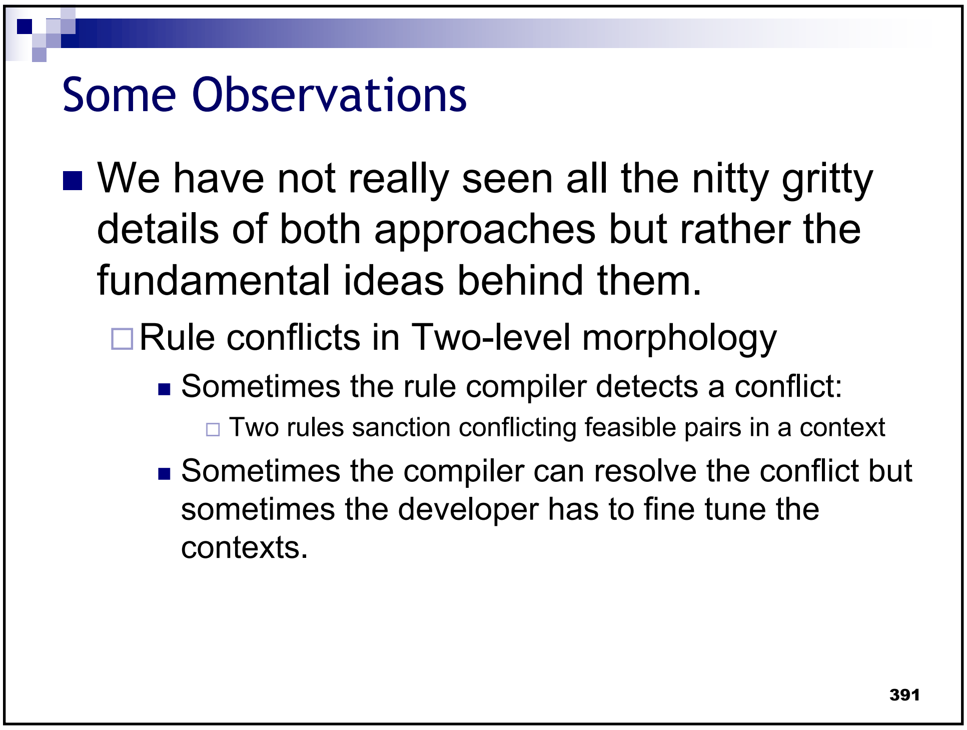
Lexicon Transducer

○

(Partial) Morphographemic Transducer

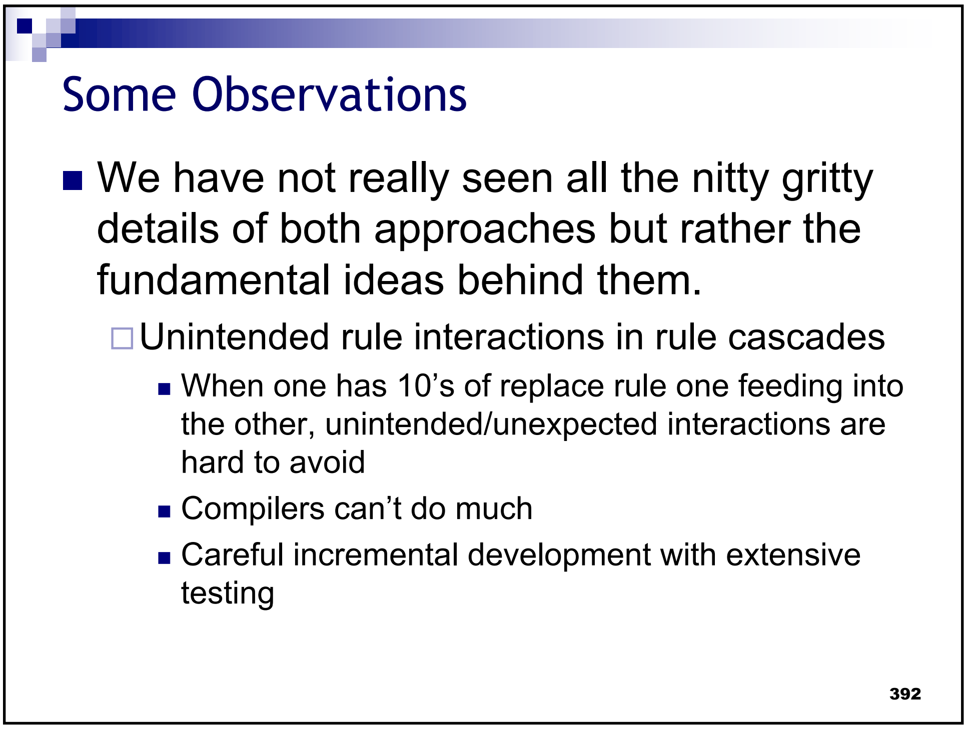
⇨

(partial) TMA



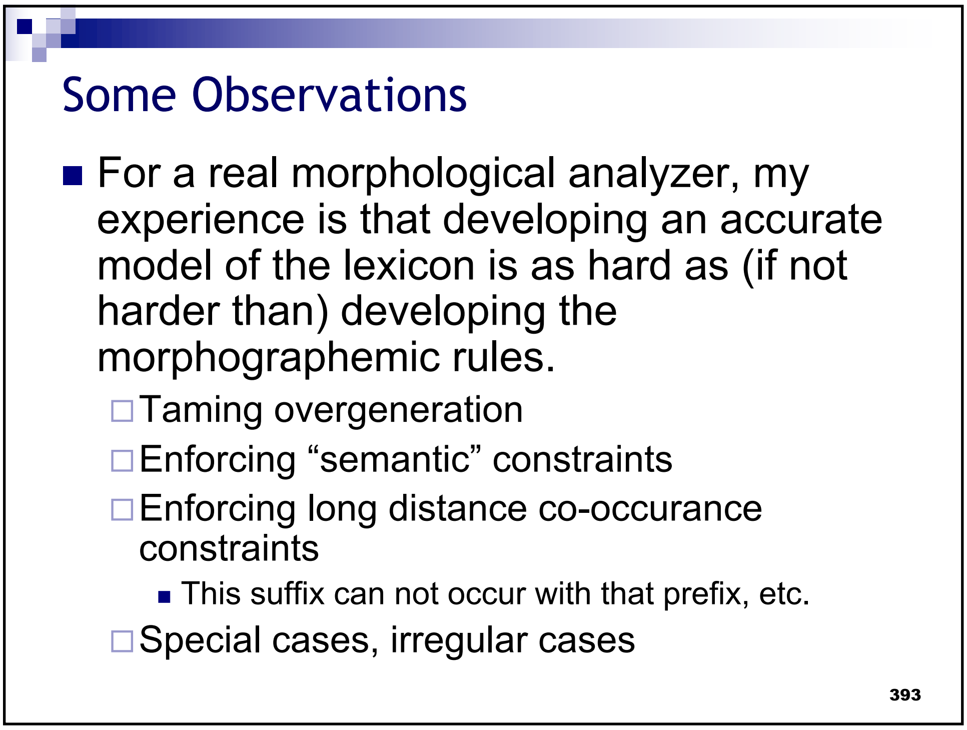
Some Observations

- We have not really seen all the nitty gritty details of both approaches but rather the fundamental ideas behind them.
  - □ Rule conflicts in Two-level morphology
    - Sometimes the rule compiler detects a conflict:
      - □ Two rules sanction conflicting feasible pairs in a context
    - Sometimes the compiler can resolve the conflict but sometimes the developer has to fine tune the contexts.



Some Observations

- We have not really seen all the nitty gritty details of both approaches but rather the fundamental ideas behind them.
  - □ Unintended rule interactions in rule cascades
    - When one has 10’s of replace rule one feeding into the other, unintended/unexpected interactions are hard to avoid
    - Compilers can’t do much
    - Careful incremental development with extensive testing



Some Observations

- For a real morphological analyzer, my experience is that developing an accurate model of the lexicon is as hard as (if not harder than) developing the morphographemic rules.
  - Taming overgeneration
  - Enforcing “semantic” constraints
  - Enforcing long distance co-occurance constraints
    - This suffix can not occur with that prefix, etc.
  - Special cases, irregular cases

393

11-411
Natural Language Processing
Language Modelling and Smoothing

Kemal Oflazer

Carnegie Mellon University in Qatar

What is a Language Model?

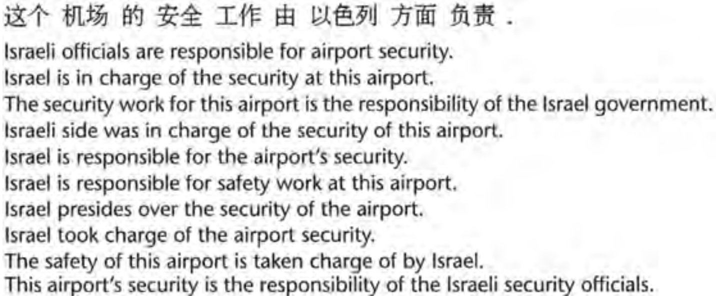
- A model that estimates how likely it is that a sequence of words belongs to a (natural) language
- Intuition
  - $p(\text{A tired athlete sleeps comfortably}) \gg p(\text{Colorless green ideas sleep furiously})$
  - $p(\text{Colorless green ideas sleep furiously}) \gg p(\text{Salad word sentence is this})$

Let’s Check How Good Your Language Model is?

- Guess the most likely next word
- The prime of his life . . . {is, was, . . . }
- The prime minister gave an . . . {ultimatum, address, expensive, . . . }
- The prime minister gave a . . . {speech, book, cheap, . . . }
- The prime number after eleven . . . {is, does, could, has, had . . . }
- The prime rib was . . . {delicious, expensive, flavorful, . . . ,} but NOT green

Where do we use a language model?

- Language models are typically used as components of larger systems.
- We’ll study how they are used later, but here’s some further motivation.
  - Speech transcription:
    - I want to learn how to wreck a nice beach.
    - I want to learn how to recognize speech.
  - Handwriting recognition:
    - I have a gub!
    - I have a gun!
  - Spelling correction:
    - We’re leaving in five minuets.
    - We’re leaving in five minutes.
  - Ranking machine translation system outputs



这个 机场 的 安全 工作 由 以色列 方面 负责 .

Israeli officials are responsible for airport security.
Israel is in charge of the security at this airport.
The security work for this airport is the responsibility of the Israel government.
Israeli side was in charge of the security of this airport.
Israel is responsible for the airport’s security.
Israel is responsible for safety work at this airport.
Israel presides over the security of the airport.
Israel took charge of the airport security.
The safety of this airport is taken charge of by Israel.
This airport’s security is the responsibility of the Israeli security officials.

Very Quick Review of Probability

- Event space (e.g., $\mathcal{X}, \mathcal{Y}$), usually discrete for the purposes of this class.
- Random variables (e.g., X, Y)
- We say "Random variable X takes value $x \in \mathcal{X}$ with probability $p(X = x)$"
    - We usually write $p(X = x)$ as $p(x)$.
- Joint probability: $p(X = x, Y = y)$
- Conditional probability: $p(X = x \mid Y = y) \; = \dfrac{p(X = x, Y = y)}{p(Y = y)}$
- This always holds

$$p(X = x, Y = y) = \underbrace{p(X = x \mid Y = y) \cdot p(Y = y)}= \underbrace{p(Y = y \mid X = x) \cdot p(X = x)}$$

- This sometimes holds: $p(X = x, Y = y) = p(X = x) \cdot p(Y = y)$
- *True* and *estimated* probability distributions.

Language Models: Definitions

- $\mathcal{V}$ is a finite set of discrete symbols (characters, words, emoji symbols, ...), $V = |\mathcal{V}|$.
- $\mathcal{V}^+$ is the infinite set of finite-length sequence of symbols from $\mathcal{V}$ whose final symbol is □.
- $p : \mathcal{V}^+ \to \mathbb{R}$ such that
  - For all $\boldsymbol{x} \in \mathcal{V}^+ p(\boldsymbol{x}) \geq 0$
  - p is a proper probability distribution: $\displaystyle\sum_{x \in \mathcal{V}^+} p(\boldsymbol{x}) = 1$
- Language modeling: Estimate p from the training set examples

$$\boldsymbol{x}_{1:n} = \langle \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n \rangle$$

- Notation going forward:
  - x a single symbol (word, character, etc.) from $\mathcal{V}$
  - $\boldsymbol{x}$ is a sequence of symbols in $\mathcal{V}^+$ as defined above. x_i is the i^{th} symbol of $\boldsymbol{x}$.
  - $\boldsymbol{x}_{1:n}$ denotes n sequences, $\langle \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n \rangle$.
    - $\boldsymbol{x}_i$ is the i^{th} sequence in $\boldsymbol{x}_{1:n}$.
    - $[\boldsymbol{x}_i]_j$ is the j^{th} symbol of the i^{th} sequence in $\boldsymbol{x}_{1:n}$.

Issues

- Why would we want to do this?
- Are the nonnegativity and sum-to-one constraints really necessary?
- Is “finite $\mathcal{V}$” realistic?

Motivation – Noisy Channel Models

- Noisy channel models are very suitable models for many NLP problems:

$$\boxed{\text{Source (Generator)}} \to Y \to \boxed{\text{Channel (Blender)}} \to X$$

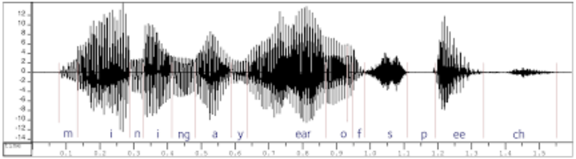
- Y is the plaintext, the true message, the missing information, the output
- X is the ciphertext, the garbled message, the observable evidence, the input
- Decoding: select the best $\boldsymbol{y}$ given $X = \boldsymbol{x}$.

$$\begin{aligned}
\hat{\boldsymbol{y}} &= \arg\max_{\boldsymbol{y}} \; p(\boldsymbol{y} \mid \boldsymbol{x}) \\
&= \arg\max_{\boldsymbol{y}} \; \frac{p(\boldsymbol{x} \mid \boldsymbol{y}) \cdot p(\boldsymbol{y})}{p(\boldsymbol{x})} \\
&= \arg\max_{\boldsymbol{y}} \quad \underbrace{p(\boldsymbol{x} \mid \boldsymbol{y})}_{\text{Channel Model}} \quad \cdot \quad \underbrace{p(\boldsymbol{y})}_{\text{Source Model}}
\end{aligned}$$

Noisy Channel Example – Speech Recognition

$\boxed{\text{Source}} \rightarrow Y$ (Seq. of words) $\rightarrow \boxed{\text{Vocal Tract}} \rightarrow X$ (Acoustic Waves)

- “Mining a year of speech” $\rightarrow$

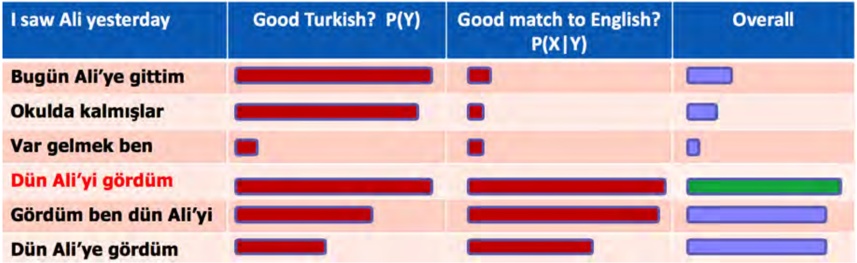


- Source model characterizes $p(\boldsymbol{y})$, “What are possible sequences of words I can say?”
- Channel model characterizes $p(\text{Acoustics} \mid \boldsymbol{y})$
  - It is hard to recognize speech
  - It is hard to wreck a nice beach
  - It is hard wreck an ice beach
  - It is hard wreck a nice peach
  - It is hard wreck an ice peach
  - It is heart to wreck an ice peach
  - . . .

Noisy Channel Example – Machine Translation

$\boxed{\text{Source}} \rightarrow Y$ (Seq. of Turkish words) $\rightarrow \boxed{\text{Translation}} \rightarrow X$ (Seq. of English Words)

- $p(\boldsymbol{x} \mid \boldsymbol{y})$ models the translation process.
- Given an observed sequence of English words, presumably generated by translating from Turkish, what is the most likely source Turkish sentence, that could have given rise to this translation?



I saw Ali yesterday	Good Turkish? P(Y)	Good match to English? P(X|Y)	Overall
Bugün Ali'ye gittim			
Okulda kalmışlar			
Var gelmek ben			
Dün Ali'yi gördüm			
Gördüm ben dün Ali'yi			
Dün Ali'ye gördüm			

Machine Transliteration

- Phonetic translation across language pairs with very different alphabets and sound system is called *transliteration*.
- *Golfbag* in English is to be transliterated to Japanese.
  - Japanese has no distinct *l* and *r* sounds - these in English collapse to the same sound. Same for English *h* and *f*.
  - Japanese uses alternating vowel-consonant syllable structure: *lfb* is impossible to pronounce without any vowels.
  - Katagana writing is based on syllabaries: different symbols for *ga*, *gi*, *gu*, etc.
  - So Golfbag is transliterated as ゴルフバッグ and pronounced as *go-ru-hu-ba-ggu*.
- So when you see a transliterated word in Japanese text, how can you find out what the English is?
  - *nyuuyooko taimuzu* → New York Times
  - *aisukuriimu* → ice-cream (and not “I scream”)
  - *ranpu* → lamp or ramp
  - *masutaazutoonamento* → Master’s Tournament

Noisy Channel Model – Other Applications

- Spelling Correction
- Grammar Correction
- Optical Character Recognition
- Sentence Segmentation
- Part-of-speech Tagging

Is finite $\mathcal{V}$ realistic?

- NO!
- We will never see all possible words in a language no matter how large the sample we look at, is.

The Language Modeling Problem

- **Input:** $\boldsymbol{x}_{1:n}$ – the “training data”.
- **Output:** $p : \mathcal{V}^{+} \rightarrow \mathbb{R}^{+}$

- p should be a “useful” measure of plausibility (not necessarily of grammaticality).

A Very Simple Language Model

- We are given $\boldsymbol{x}_{1:n}$ as the training data
  - Remember that each $\boldsymbol{x}_i$ is a sequence of symbols, that is, a "sentence"
  - So we have n sentences, and we count how many times the sentence $\boldsymbol{x}$ appears
- $p(\boldsymbol{x})$ is estimated as

$$p(\boldsymbol{x}) = \frac{|\{i : \boldsymbol{x}_i = \boldsymbol{x}\}|}{n} = \frac{c_{\boldsymbol{x}_{1:n}}(\boldsymbol{x})}{n}$$

- So we only know about n sentences and nothing else!

- What happens when you want to assign a probability to some $\boldsymbol{x}$ that is not in the training set?
- Is there a way out?

Chain Rule to the Rescue

- We break down $p(\boldsymbol{x})$ mathematically

$$
\begin{aligned}
p(\boldsymbol{X} = \boldsymbol{x}) \quad = \quad & p(X_1 = x_1) \times \\
& \quad p(X_2 = x_2 \mid X_1 = x_1) \times \\
& \quad p(X_3 = x_3 \mid \boldsymbol{X}_{1:2} = \boldsymbol{x}_{1:2}) \times \\
& \quad \vdots \\
& \quad p(X_\ell = \square) \mid \boldsymbol{X}_{1:\ell-1} = \boldsymbol{x}_{1:\ell-1}) \\
\\
= \quad & \prod_{j=1}^{\ell} p(X_j = x_j \mid \boldsymbol{X}_{1:j-1} = \boldsymbol{x}_{1:j-1})
\end{aligned}
$$

- This is an exact formulation.
- Each word is conditioned on all the words coming before it!

Approximating the Chain Rule Expansion – The Unigram Model

$$
\begin{aligned}
p(\boldsymbol{X} = \boldsymbol{x}) &= \prod_{j=1}^{\ell} p(X_j = x_j \mid \boldsymbol{X}_{1:j-1} = \boldsymbol{x}_{1:j-1}) \\
&\overset{assumption}{=} \prod_{j=1}^{\ell} p_\theta(X_j = x_j) = \prod_{j=1}^{\ell} \theta_{x_j} \approx \prod_{j=1}^{\ell} \hat{\theta}_{x_j}
\end{aligned}
$$

- $\hat{\theta}'$s are maximum likelihood estimates:

$$\forall v \in \mathcal{V}, \quad \hat{\theta}_v = \frac{|(i,j) : [\boldsymbol{x}_i]_j = v|}{N} = \frac{c_{\boldsymbol{x}_{1:n}}(v)}{N}$$

- $N = \sum_{i=1}^{n} |\boldsymbol{x}_i|$
- This is also known as “relative frequency estimation”.
- The unigram model is also known as the “bag of words” model. Why?

Unigram Models – The Good and the Bad

Pros:

- Easy to understand
- Cheap
  - Not many parameters
  - Easy to compute
- Good enough for maybe information retrieval

Cons:

- “Bag of Words” assumption is not linguistically accurate.
  - $$p(\text{the the the the}) \gg p(\text{I want to run})$$
- “Out of vocabulary” problem.
  - What happens if you encounter a word you never saw before?

- Generative Process: keep on randomly picking words until you pick □.
- We really never use unigram models!

Approximating the Chain Rule Expansion – Markov Models

Markov Models $\equiv$ n-gram Models

$$
\begin{aligned}
p(\boldsymbol{X}=\boldsymbol{x}) &= \prod_{j=1}^{\ell} p(X_j = x_j \mid \boldsymbol{X}_{1:j-1} = \boldsymbol{x}_{1:j-1}) \\
&\overset{assumption}{=} \prod_{j=1}^{\ell} p_\theta(X_j = x_j \mid \underbrace{\boldsymbol{X}_{j-n+1:j-1} = \boldsymbol{x}_{j-n+1:j-1})}_{\text{last } n-1 \text{ words}}
\end{aligned}
$$

- n-gram models $\equiv (n-1)^{th}$-order Markov assumption.
- Unigram model model with when $n = 1$
- Trigram models $(n = 3)$ are widely used.
- 5-gram models $(n = 5)$ are quite common in statistical machine translation.

Estimating n-gram Models

	unigram	bigram	trigram	general n-gram
$p_\theta(\boldsymbol{x}) =$	$\prod_{j=1}^{\ell} \theta_{x_j}$	$\prod_{j=1}^{\ell} \theta_{x_j \mid x_{j-1}}$	$\prod_{j=1}^{\ell} \theta_{x_j \mid x_{j-2} x_{j-1}}$	$\prod_{j=1}^{\ell} \theta_{x_j \mid \boldsymbol{x}_{j-n+1:j-1}}$
Parameters:	θ_v $\forall v \in \mathcal{V}$	$\theta_{v \mid v'}$ $\forall v' \in \mathcal{V},$ $\forall v \in \mathcal{V} \cup \{\square\}$	$\theta_{v \mid v'' v'}$ $\forall v', v'' \in \mathcal{V},$ $\forall v \in \mathcal{V} \cup \{\square\}$	$\theta_{v \mid \boldsymbol{h}}$ $\forall \boldsymbol{h} \in \mathcal{V}^{n-1},$ $\forall v \in \mathcal{V} \cup \{\square\}$
MLE:	$\frac{c(v)}{N}$	$\frac{c(v'v)}{c(v')}$	$\frac{c(v''v'v)}{c(v''v')}$	$\frac{c(\boldsymbol{h}v)}{c(\boldsymbol{h})}$

The Problem with MLE

- The curse of dimensionality: the number of parameters grows exponentially in n.
- Data sparseness: most n-grams will never be observed – even when they are linguistically plausible.
- What is the probability of unseen words? (0 ?)
- But that's not what you want. Test set will usually include words not in training set.
  - What is $p(\text{Nebuchadnezzur} \mid \text{son of})$?
- A single 0 probability will set the estimate to 0. Not acceptable!

Engineering Issues – Log Probabilities

- Note that computation of $p_\theta(\boldsymbol{x})$ involves multiplication of numbers each of which are between 0 and 1.
- So multiplication hits *underflow*: computationally the product can not be represented or computed.
- In implementation, probabilities are represented by the *logarithms* (between $-\infty$ and 0) and multiplication is replaced by addition.

Dealing with Out-of-Vocabulary Words

- Quick and dirty approach
  - Decide what is in the vocabulary (e.g., all words with frequency $>$ say 10).
  - Add `UNK` to the vocabulary.
  - Replace all unknown words with `UNK`
  - Estimate as usual.
- Build a language model at the character level.
  - What are advantages and disadvantages?

Smoothing Language Models

- We can not have 0 probability n-grams. So we should shave off some probability mass from seen n-grams to give to unseen n-grams.
  - The Robin-Hood Approach – steal some probability from the *have*s to *have-not*s.
- Simplest method: Laplace Smoothing
- Interpolation
- Stupid backoff.
- Long-standing best method: modified Kneser-Ney smoothing

Laplace Smoothing

- We add 1 to all counts! So words with 0 counts will be assumed to have count 1.
  - Unigram probabilities: $p(v) = \dfrac{c(v) + 1}{N + V}$
  - Bigram probabilities: $p(v \mid v') = \dfrac{c(v'v) + 1}{c(v') + V}$
- One can also use *Add-k* smoothing for some fractional $k, 0 < k \leq 1)$
- It turns out this method is very simple but shaves off too much of the probability mass. (See book for an example.)

Interpolation

- We estimate n-gram probabilities by combining count-based estimates from n- and lower grams.

$$\hat{p}(v \mid v''v') = \lambda_1 p(v \mid v''v') + \lambda_2 p(v \mid v') + \lambda_3 p(v)$$

- $\sum_i \lambda_i = 1$
- λ's are estimated by maximizing the likelihood of a *held-out* data.

Stupid Backoff

- Gives up the idea of making the language model a true probability distribution.
- Works quite well with very large training data (e.g. web scale) and large language models
- If a given n-gram has never been observed, just use the next lower gram’s estimate scaled by a fixed weight λ (terminates when you reach the unigram)

Kneser-Ney Smoothing

- Kneser-Ney smoothing and its variants (interpolated Kenser-Ney or modified Kneser-Ney) use absolute discounting.
- The math is a bit more involved. See the book if you are interested.

Toolkits

- These days people build language models using well-established toolkits:
  - SRILM Toolkit (`https://www.sri.com/engage/products-solutions/sri-language-modeling-toolkit`)
  - CMU Statistical Language Modeling Toolkit (`http://www.speech.cs.cmu.edu/SLM_info.html`)
  - KenLM Language Model Toolkit (`https://kheafield.com/code/kenlm/`)
- Each toolkit provides executables and/or API and options to build, smooth, evaluate and use language models. See their documentation.

n-gram Models– Assessment

Pros:

- Easy to understand
- Cheap (with modern hardware/memory)
- Good enough for machine translation, speech recognition, contextual spelling correction, etc.

Cons:

- Markov assumption is not linguistically accurate.
  - but not as bad as unigram models
- “Out of vocabulary” problem.

Evaluation – Language Model Perplexity

- Consider language model that assigns probabilities to a sequence of digits (in speech recognition)
- Each digit occurs with the same probability $p = 0.1$
- Perplexity for a sequence of N digits $D = d_1 d_2 \cdots d_n$ is

$$
\begin{aligned}
PP(D) &\overset{def}{=} p(d_1 d_2 \cdots d_n)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{p(d_1 d_2 \cdots d_n)}} \\
&= \sqrt[N]{\frac{1}{\prod_{i=1}^{N} p(d_i)}} \\
&= \sqrt[N]{\frac{1}{(\frac{1}{10})^N}} \\
&= 10
\end{aligned}
$$

- How can we interpret this number?

Evaluation – Language Model Perplexity

- Intuitively, language models should assign high probability to “real language” they have not seen before.
- Let $\bar{\boldsymbol{x}}_{1:m}$ be a sequence of m sentences, that we have not seen before (held-out or test set)
- Probability of $\bar{\boldsymbol{x}}_{1:m} = \prod_{i=1}^{m} p(\bar{\boldsymbol{x}}_i) \Rightarrow$ Log probability of $\bar{\boldsymbol{x}}_{1:m} = \sum_{i=1}^{m} \log_2 p(\bar{\boldsymbol{x}}_i)$
- Average log probability of per word of $\bar{\boldsymbol{x}}_{1:m}$ is:

$$l = \frac{1}{M} \sum_{i=1}^{m} \log_2 p(\bar{\boldsymbol{x}}_i)$$

where $M = \sum_{i=1}^{m} |\bar{\boldsymbol{x}}_i|$

- Perplexity relative to $\bar{\boldsymbol{x}}_{1:m} \overset{def}{=} 2^{-l}$
  - Intuitively, perplexity is average “confusion” after each word. Lower is better!

Understanding Perplexity

- $2^{-\frac{1}{M}\sum_{i=1}^{m}\log_2 p(\overline{\vec{x}_i})}$ is really a branching factor.
- Assign probability of 1 to the test data $\Rightarrow$ perplexity $= 1$. No confusion.
- Assign probability of $\frac{1}{V}$ to each word $\Rightarrow$ perplexity $= V$. Equal confusion after each word!
- Assign probability of 0 to anything $\Rightarrow$ perplexity $= \infty$
  - We really should have for any $x \in \mathcal{V}^{+} p(\boldsymbol{x}) > 0$

Entropy and Cross-entropy

- Suppose that there are eight horses running in an upcoming race.
- Your friend is on the moon.
- It’s really expensive to send a bit to the moon!
- You want to send him the results.
-

Clinton	000	Huckabee	100
Edwards	001	McCain	101
Kucinich	010	Paul	110
Obama	011	Romney	111

- Expected number of bits to convey a message is 3 bits.

Entropy and Cross-entropy

- Suppose the probabilities over the outcome of the race are not at all even.
-

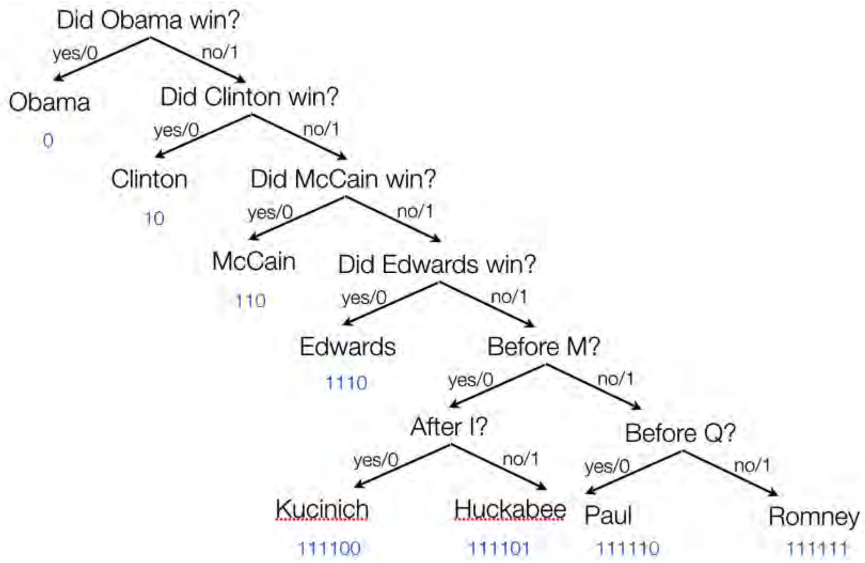
Clinton	1/4	Huckabee	1/64
Edwards	1/16	McCain	1/8
Kucinich	1/64	Paul	1/64
Obama	1/2	Romney	1/64

- You can encode the winner using the following coding scheme

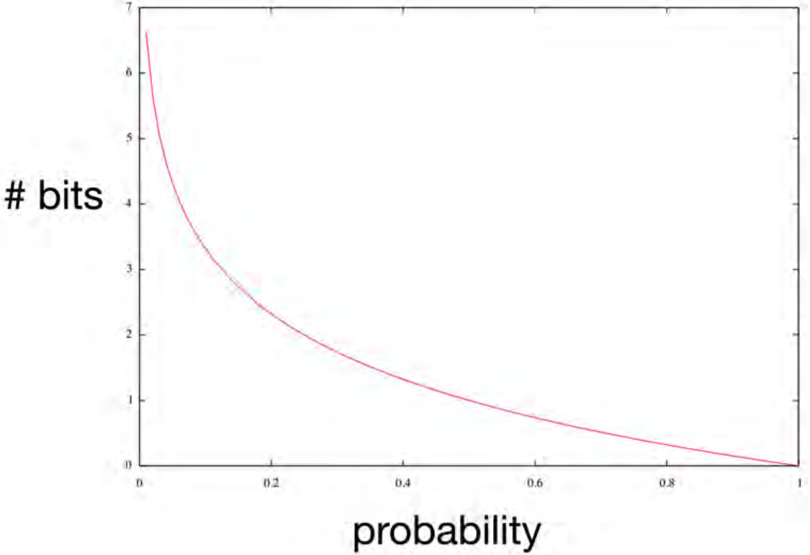
Clinton	10	Huckabee	111101
Edwards	1110	McCain	110
Kucinich	11110	Paul	111110
Obama	0	Romney	111111

- How did we get these codes?

Another View



Bits vs Probabilities



Entropy

- Entropy of a Distribution

$$H(p) = -\sum_{x\in\mathcal{X}} p(x)\log p(x)$$

- Always ≥ 0 and maximal when p is uniform.

$$H(p_{uniform}) = -\sum_{x\in\mathcal{X}} \frac{1}{|\mathcal{X}|}\log\frac{1}{|\mathcal{X}|} = \log|\mathcal{X}|$$

Cross-entropy

- Cross-entropy uses one distribution to tell us something about another distribution.

$$H(p;q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x)$$

- The difference $H(p;q) - H(p)$ tells us how many extra bits (on average) we waste by using q instead of p.
- Extra bits make us sad; we can therefore think of this as a measure of regret.
- We want to choose q so that $H(p;q)$ is small.
- Cross-entropy is an estimate of of the average code-length (bits per message) when using q as a proxy for p.

Cross-entropy and Betting

- Before the horse race, place your bet.
- Regret is how sad you feel after you find out who won.
- What's your average score after you place your bets and test many times?
- Upper bound on regret: uniform betting
- Lower bound on regret: proportional betting on the true distribution for today's race.
- The better our estimated distribution is, the closer we get to the lower bound (lower regret)!

How does this Relate to Language Models?

- p_{train}: training sample (which horses we have seen before)
- p_{test}: test sample (which horse will win today)
- q: our (estimated) model (or code)
- Real goal when training: make $H(p_{test}; q)$ small.
- We don't know p_{test}! The closest we have is p_{train}.
- So make $H(p_{train}; q)$small.
- But that overfits and can lead to infinite regret.
- Smoothing hopefully makes q more like p_{test}.

What do n-gram Models Know?

- They (sort of) learn:
    - Rare vs. common words and patterns
    - Local syntax (an elephant, a rhinoceros)
    - Words with related meanings (ate apples)
    - Punctuation and spelling
- They have no idea about:
    - Sentence structure
    - Underlying rules of agreement/spelling/etc.
    - Meaning
    - The World

Unigram Model Generation

```
first, from less the This different 2004), out which goal 19.2 Model
their It ˜(i?1), given 0.62 these (x0; match 1 schedule. x 60
1998. under by Notice we of stated CFG 120 be 100 a location accuracy
If models note 21.8 each 0 WP that the that Novak. to function; to
[0, to different values, model 65 cases. said -24.94 sentences not
that 2 In to clustering each K&M 100 Boldface X))] applied; In 104
S. grammar was (Section contrastive thesis, the machines table -5.66
trials: An the textual (family applications.We have for models 40.1 no
156 expected are neighborhood
```

Bigram Model Generation

e. (A.33) (A.34) A.5 ModelS are also been completely surpassed in performance on drafts of online algorithms can achieve far more so while substantially improved using CE. 4.4.1 MLEasaCaseofCE 71 26.34 23.1 57.8 K&M 42.4 62.7 40.9 44 43 90.7 100.0 100.0 100.0 15.1 30.9 18.0 21.2 60.1 undirected evaluations directed DEL1 TRANS1 neighborhood. This continues, with supervised init., semisupervised MLE with the METU-Sabanci Treebank 195 ADJA ADJD ADV APPR APPRART APPO APZR ART CARD FM ITJ KOUI KOUS KON KOKOM NN NN NN IN JJ NN Their problem is y x. The evaluation offers the hypothesized link grammar with a Gaussian

Trigram Model Generation

top(xI ,right,B). (A.39) vine0(X, I) rconstit0(I 1, I). (A.40) vine(n). (A.41) These equations were presented in both cases; these scores u<AC>into a probability distribution is even smaller(r =0.05). This is exactly fEM. During DA, is gradually relaxed. This approach could be efficiently used in previous chapters) before training (test) K&MZeroLocalrandom models Figure4.12: Directed accuracy on all six languages. Importantly, these papers achieved state-of-the-art results on their tasks and unlabeled data and the verbs are allowed (for instance) to select the cardinality of discrete structures, like matchings on weighted graphs (McDonald et al., 1993) (35 tag types, 3.39 bits). The Bulgarian,

The Trade-off

- As we increase n, the stuff the model generates looks better and better, and the model gives better probabilities to the training data.
- But as n gets big, we tend toward the history model, which has a lot of zero counts and therefore isn't helpful for data we haven't seen before.
- Generalizing vs. Memorizing

11-411
Natural Language Processing
Classification

Kemal Oflazer

Carnegie Mellon University in Qatar

Text Classification

- We have a set of documents (news items, emails, product reviews, movie reviews, books, . . .)
- Classify this set of documents into a small set *classes.*
- Applications:
  - Topic of a news article (classic example: finance, politics, sports, . . .)
  - Sentiment of a movie or product review (good, bad, neutral)
  - Email into spam or not or into a category (business, personal, bills, . . .)
  - Reading level (K-12) of an article or essay
  - Author of a document (Shakespeare, James Joyce, . . .)
  - Genre of a document (report, editorial, advertisement, blog, . . .)
  - Language identification

Notation and Setting

- We have a set of n documents (texts) $\boldsymbol{x_i} \in \mathcal{V}^+$
  - We assume the texts are segmented already.
- We have set $\mathcal{L}$ of labels, ℓ_i
- Human experts annotate documents with labels and give us $\{(\boldsymbol{x_1}, \ell_1), (\boldsymbol{x_2}, \ell_2), \cdots, (\boldsymbol{x_n}, \ell_n)\}$
- We learn a *classifier* `classify` : $\mathcal{V}^+ \to \mathcal{L}$ with this labeled training data.
- Afterwards, we use `classify` to classify new documents into their classes.

Evaluation

- Accuracy:

$$\mathsf{A}(\texttt{classify}) = \sum_{\substack{\boldsymbol{x}\in\mathcal{V}^{+},\ell\in\mathcal{L},\\ \texttt{classify}(\boldsymbol{x})=\ell}} p(\boldsymbol{x}, \ell)$$

  where p is the true distribution over data. Error is $1 - \mathsf{A}$.

- This is estimated using a test set $\{(\bar{\boldsymbol{x}}_1, \bar{\ell}_1), (\bar{\boldsymbol{x}}_2, \bar{\ell}_2), \cdots, (\bar{\boldsymbol{x}}_m, \bar{\ell}_m)\}$

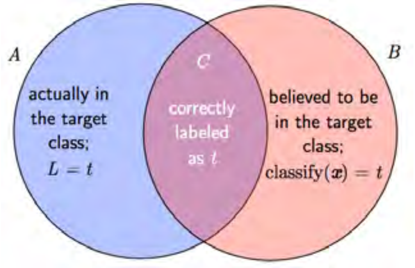
$$\hat{\mathsf{A}}(\texttt{classify}) = \frac{1}{m}\sum_{i=1}^{m} 1\{\texttt{classify}(\bar{\boldsymbol{x}}_i) = \bar{\ell}_i\}$$

Issues with Using Test Set Accuracy

- Class imbalance: if $p(L = \text{not spam}) = 0.99$, then you can get Â ≈ 0.99 by always guessing “not spam”
- Relative importance of classes or cost of error types.
- Variance due to the test data.

Evaluation in the Two-class case

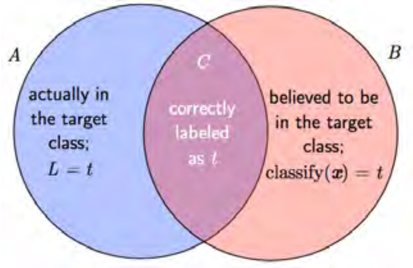
- Suppose we have one of the classes $t \in \mathcal{L}$ as the target class.
- We would like to identify documents with label t in the test data.
  - Like information retrieval
- We get



A
B
C
actually in the target class; $L = t$
correctly labeled as t
believed to be in the target class; $\text{classify}(\boldsymbol{x}) = t$

- Precision $\hat{P} = \dfrac{C}{B}$ (percentage of documents `classify` correctly labeled as t)
- Recall $\hat{R} = \dfrac{C}{A}$ (percentage of actual t labeled documents correctly labeled as t)
- $F_1 = 2\dfrac{\hat{P} + \hat{R}}{\hat{P} \cdot \hat{R}}$

A Different View – Contingency Tables


A
C
B
actually in the target class; $L = t$
correctly labeled as t
believed to be in the target class; $\text{classify}(\boldsymbol{x}) = t$

	$L = t$	$L \neq t$	
$\texttt{classify}(\boldsymbol{X}) = t$	C(true positives)	$B \backslash C$(false positives)	B
$\texttt{classify}(\boldsymbol{X}) \neq t$	$C \backslash A$(false negatives)	(true negatives)	
	A		

Evaluation with > 2 Classes

- **Macroaveraged precision and recall**: let each class be the target and report the average $\hat{P}$ and $\hat{R}$ across all classes.
- **Microaveraged precision and recall**: pool all one-vs.-rest decisions into a single contingency table, calculate $\hat{P}$ and $\hat{R}$ from that.

Cross-validation

- Remember that $\hat{A}$, $\hat{P}$, $\hat{R}$, and $\hat{F}_1$ are all estimates of the classifier's quality under the true data distribution.
  - Estimates are noisy!
- K-fold cross validation
  - Partition the training data into K nonoverlapping "folds", $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^K$,
  - For $i \in \{1, \ldots, K\}$
    - Train on $\boldsymbol{x}_{1:n} \backslash \boldsymbol{x}^i$, using $\boldsymbol{x}^i$ as development data
    - Estimate quality on the $\boldsymbol{x}^i$ development set as $\hat{\text{A}}^i$
  - Report average accuracy as $\hat{\text{A}} = \frac{1}{K} \sum_{i=1}^{K} \hat{\text{A}}^i$ and perhaps also the standard deviation.

Features in Text Classification

- Running example $\boldsymbol{x}$ = “The spirit is willing but the flesh is weak”
- *Feature random variables*
- For $j \in \{1, \ldots, d\}$ F_j is a discrete random variable taking values in $\mathcal{F}_j$
- Most of the time these can be frequencies of words or n-grams in a text.
  - $f_{f-spirit} = 1, f_{f-is} = 2, f_{f-the-flesh} = 1, \ldots$
- They can be boolean “exists” features.
  - $f_{e-spirit} = 1, f_{e-is} = 1, f_{f-strong} = 0, \ldots$

Spam Detection

- A training set of email messages (marked *Spam* or *Not-Spam*)
- A set of features for each message (considered as a bag of words)
  - For each word: Number of occurrences
  - Whether phrases such as “Nigerian Prince”, “email quota full”, “won ONE HUNDRED MILLION DOLLARS” are in the message
  - Whether it is from someone you know
  - Whether it is reply to your message
  - Whether it is from your domain (e.g., `cmu.edu`)

Movie Ratings

- A training set of movie reviews (with star ratings 1 - 5)
- A set of features for each message (considered as a bag of words)
  - For each word: Number of occurrences
  - Whether phrases such as *Excellent*, *sucks*, *blockbuster*, *biggest*, *Star Wars*, *Disney*, *Adam Sandler*, . . . are in the review

Probabilistic Classification

- Documents are preprocessed: each document $\boldsymbol{x}$ is mapped to a d-dimensional feature vector $\boldsymbol{f}$.
- Classification rule

$$
\begin{aligned}
\text{classify}(\boldsymbol{f}) &= \arg\max_{\ell \in \mathcal{L}} \; p(\ell \mid \boldsymbol{f}) \\
&= \arg\max_{\ell \in \mathcal{L}} \; \frac{p(\ell,\boldsymbol{f})}{p(\boldsymbol{f})} \\
&= \arg\max_{\ell \in \mathcal{L}} \; p(\ell,\boldsymbol{f}) \text{(Why?)}
\end{aligned}
$$

Naive Bayes Classifier

$$p(L = \ell, F_1 = f_1, \ldots, F_d = f_d) = p(\ell) \prod_{j=1}^{d} p(F_j = f_j \mid \ell)$$

$$= \pi_\ell \prod_{j=1}^{d} \theta_{f_j|j,\ell}$$

- Parameters: π_ℓ is the class or label prior.
  - The probability that a document belongs to class ℓ – without considering any of its features.
  - They can be computed directly from the training data $\{(\boldsymbol{x_1}, \ell_1), (\boldsymbol{x_2}, \ell_2), \cdots, (\boldsymbol{x_n}, \ell_n)\}$. These sum to 1.
- For each feature function j and label ℓ, a distribution over values $\theta_{*|j,\ell}$
  - These sum to 1 for every (j, ℓ) pair.

Generative vs Discriminative Classifier

- Naive Bayes is known as a *Generative classifier*.
- Generative Classifiers build a model of each class.
- Given an observation (document), they return the class most likely have generated that observation.

- A *discriminative classifier* instead learns what features from the input are useful to discriminate between possible classes.

The Most Basic Naive Bayes Classifier

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

fairy always love to it
it whimsical it I
and seen are anyone
friend happy dialogue
adventure recommend
who sweet of satirical
movie it
it I but to romantic I
several yet
again it the humor
the seen would
to scenes I the manages
the
fun I and times and
about while
whenever have
conventions
with

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

The Most Basic Naive Bayes Classifier

- Features are just words x_j in $\boldsymbol{x}$
- **Naive Assumption**: Word positions do not matter – “bag of words”.
- **Conditional Independence**: Feature probabilities $p(x_i \mid \ell)$ are independent given the class ℓ.
  - $p(\boldsymbol{x} \mid \ell) = \prod_{j=1}^{|\boldsymbol{x}|} p(x_j \mid \ell)$
  - The probability that a word in a sports document is “soccer” is estimated as $p(\text{soccer} \mid \text{sports})$ by counting “soccer” in all sports documents.
- So

$$\text{classify}(\boldsymbol{x}) = \arg\max_{\ell \in \mathcal{L}} \ \pi_\ell \prod_{j=1}^{|\boldsymbol{x}|} p(x_j \mid \ell)$$

- Smoothing is very important as any new document may have unseen words.

The Most Basic Naive Bayes Classifier

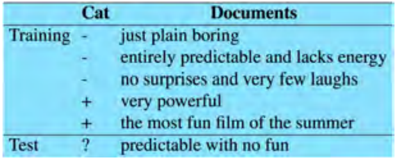
$$\text{classify}(\boldsymbol{x}) = \arg\max_{\ell \in \mathcal{L}} \; \pi_\ell \prod_{j=1}^{|\boldsymbol{x}|} p(x_j \mid \ell)$$

$$\Downarrow$$

$$\text{classify}(\boldsymbol{x}) = \arg\max_{\ell \in \mathcal{L}} \; \left(\log \pi_\ell + \sum_{j=1}^{|\boldsymbol{x}|} \log p(x_j \mid \ell) \right)$$

- All computations are done in $\log$ space to avoid underflow and increase speed.
- Class prediction is based on a linear combination of the inputs.
- Hence Naive Bayes is confidered as a *linear classifier*.

An Example



	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

- $|V| = 20$
- Add 1 Laplace smoothing

$$\pi_- = p(-) = \frac{3}{5} \quad \pi_+ = p(+) = \frac{2}{5}$$

$$N_- = 14 \quad N_+ = 9$$

$$p(\text{“predictable”} \mid -) = \frac{1+1}{14+20} \quad p(\text{“predictable”} \mid +) = \frac{0+1}{9+20}$$

$$p(\text{“no”} \mid -) = \frac{1+1}{14+20} \quad p(\text{“no”} \mid +) = \frac{0+1}{9+20}$$

$$p(\text{“fun”} \mid -) = \frac{0+1}{14+20} \quad p(\text{“fun”} \mid +) = \frac{1+1}{9+20}$$

$$p(+)p(\boldsymbol{s} \mid +) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

$$p(-)p(\boldsymbol{s} \mid -) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

Other Optimizations for Sentiment Analysis

- Ignore unknown words in the test.
- Ignore **stop words** like *the*, *a*, *with*, etc.
  - Remove most frequent 10-100 words from the training and test documents.
- Count vs existence of words: Binarized features.
- Negation Handling `didnt like this movie , but I` → `didnt NOT_like NOT_this NOT_movie , but I`

Formulation of a Discriminative Classifier

- A discriminative model computes $p(\ell \mid \boldsymbol{x})$ to discriminate among different values of ℓ, using combinations of d features of $\boldsymbol{x}$.

$$\hat{\ell} = \underset{\ell \in \mathcal{L}}{\arg\max}\; p(\ell \mid \boldsymbol{x})$$

- There is no obvious way to map features to probabilities.
- Assuming features are *binary-valued* and they are both functions of $\boldsymbol{x}$ and class ℓ we can write

$$p(\ell \mid \boldsymbol{x}) = \frac{1}{Z} exp\left(\sum_{i=1}^{d} w_i f_i(\ell, \boldsymbol{x})\right)$$

  where Z is the normalization factor to make everything a probability and w_i are weights for features.
- $p(\ell \mid \boldsymbol{x})$ can be then be formally defined with normalization as

$$p(\ell \mid \boldsymbol{x}) = \frac{exp\left(\sum_{i=1}^{d} w_i f_i(\ell, \boldsymbol{x})\right)}{\sum_{\ell' \in \mathcal{L}} exp\left(\sum_{i=1}^{d} w_i f_i(\ell', \boldsymbol{x})\right)}$$

Some Features

- Remember features are *binary-valued* and are both functions of $\boldsymbol{x}$ and class ℓ.
- Suppose we are doing sentiment classification. Here are some sample feature functions:
  - $f_1(\ell, \boldsymbol{x}) = \begin{cases} 1 & \text{if "great"} \in \boldsymbol{x} \ \&\ \ell = + \\ 0 & \text{otherwise} \end{cases}$
  - $f_2(\ell, \boldsymbol{x}) = \begin{cases} 1 & \text{if "second-rate"} \in \boldsymbol{x} \ \&\ \ell = - \\ 0 & \text{otherwise} \end{cases}$
  - $f_3(\ell, \boldsymbol{x}) = \begin{cases} 1 & \text{if "no"} \in \boldsymbol{x} \ \&\ \ell = + \\ 0 & \text{otherwise} \end{cases}$
  - $f_4(\ell, \boldsymbol{x}) = \begin{cases} 1 & \text{if "enjoy"} \in \boldsymbol{x} \ \&\ \ell = - \\ 0 & \text{otherwise} \end{cases}$

Mapping to a Linear Formulation

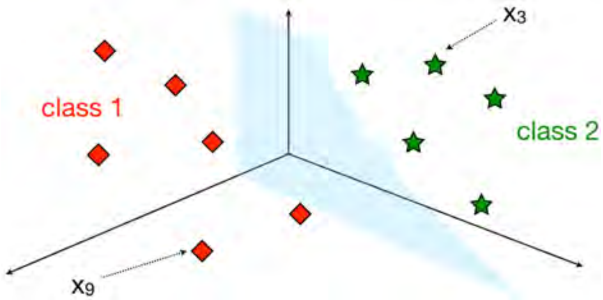
- If the goal is just classification, the denominator can be ignored

$$
\begin{aligned}
\hat{\ell} &= \arg\max_{\ell \in \mathcal{L}} p(\ell \mid \boldsymbol{x}) \\
&= \arg\max_{\ell \in \mathcal{L}} \frac{exp\left(\sum_{i=1}^{d} w_i f_i(\ell,\boldsymbol{x})\right)}{\sum_{\ell' \in \mathcal{L}} exp\left(\sum_{i=1}^{d} w_i f_i(\ell',\boldsymbol{x})\right)} \\
&= \arg\max_{\ell \in \mathcal{L}} exp\left(\sum_{i=1}^{d} w_i f_i(\ell,\boldsymbol{x})\right) \\
\hat{\ell} &= \arg\max_{\ell \in \mathcal{L}} \sum_{i=1}^{d} w_i f_i(\ell,\boldsymbol{x})
\end{aligned}
$$

- Thus we have a linear combination of features for decision making.

Two-class Classification with Linear Models

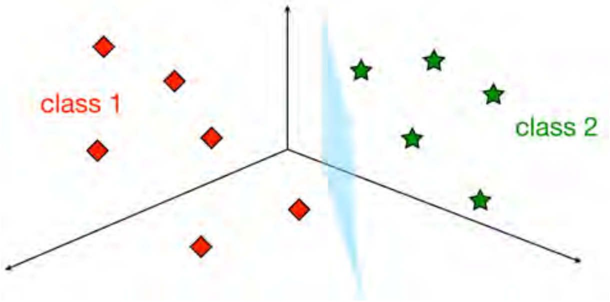
- Big idea: "map" a document $\boldsymbol{x}$ into a d-dimensional (feature) vector $\boldsymbol{\Phi}(\boldsymbol{x})$, and learn a hyperplane defined by vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_d]$.
- Linear decision rule:
  - Decide on class 1 if $\boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}) > 0$
  - Decide on class 2 if $\boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}) \leq 0$



- Parameters are $\boldsymbol{w} \in \mathbb{R}^d$. They determine the separation hyperplane.

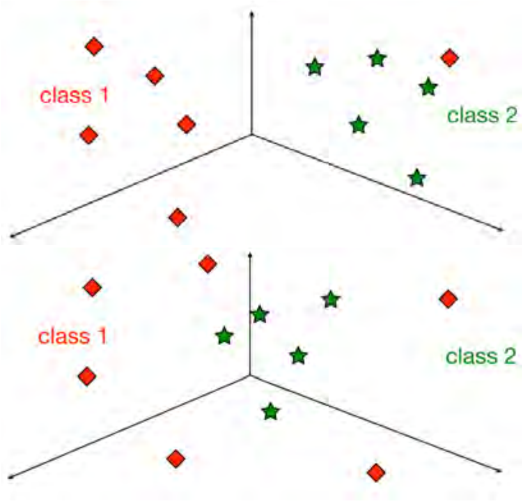
Two-class Classification with Linear Models

- There may be more than one separation hyperplane.



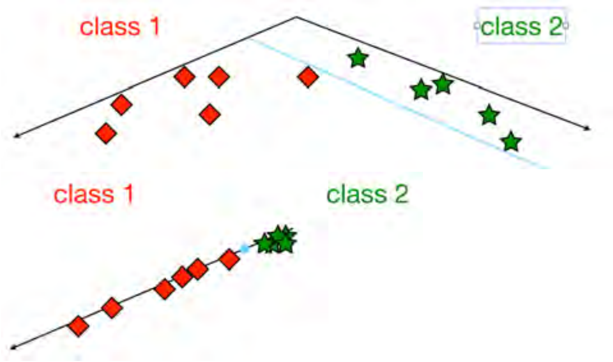
Two-class Classification with Linear Models

- There may not be a separation hyperplane. The data is not linearly separable!



Two-class Classification with Linear Models

- Some features may not be actually relevant.



The Perceptron Learning Algorithm for Two Classes

- A very simple algorithm guaranteed to eventually find a linear separator hyperplane (determine $\boldsymbol{w}$), if one exists.
- If one doesn't, the perceptron will oscillate!
- Assume our classifier is

$$\texttt{classify}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}) > 0 \\ 0 & \text{if } \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}) \leq 0 \end{cases}$$

- Start with $\boldsymbol{w} = \boldsymbol{0}$
- for $t = 1, \ldots, T$
  - $i = t \mod N$
  - $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \left(\ell_i - \texttt{classify}(\boldsymbol{x}_i)\right) \boldsymbol{\Phi}(\boldsymbol{x}_i)$
- Return $\boldsymbol{w}$
- α is the *learning rate* – determined by experimentation.

Linear Models for Classification

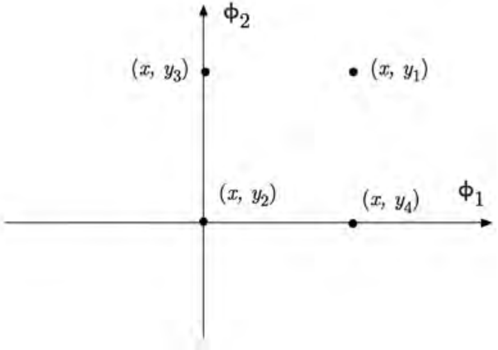
- Big idea: "map" a document $\boldsymbol{x}$ into a d-dimensional (feature) vector $\boldsymbol{\Phi}(\boldsymbol{x}, \ell)$, and learn a hyperplane defined by vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_d]$.
- Linear decision rule

$$\texttt{classify}(\boldsymbol{x}) = \hat{\ell} = \arg\max_{\ell \in \mathcal{L}} \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \ell)$$

  where $\boldsymbol{\Phi} : \mathcal{V}^{+} \times \mathcal{L} \rightarrow \mathbb{R}^{d}$
- Parameters are $\boldsymbol{w} \in \mathbb{R}^{d}$.

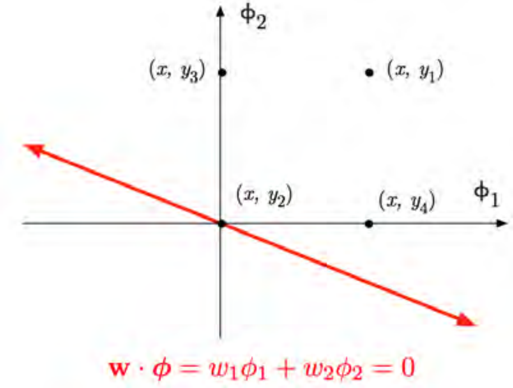
A Geometric View of Linear Classifiers

- Suppose we have an instance of $\boldsymbol{w}$ and $\mathcal{L} = \{y_1, y_2, y_3, y_4\}$.
- We have two simple binary features ϕ_1, and ϕ_2
- $\boldsymbol{\Phi}(\boldsymbol{x}, \ell)$ are as follows:



A Geometric View of Linear Classifiers

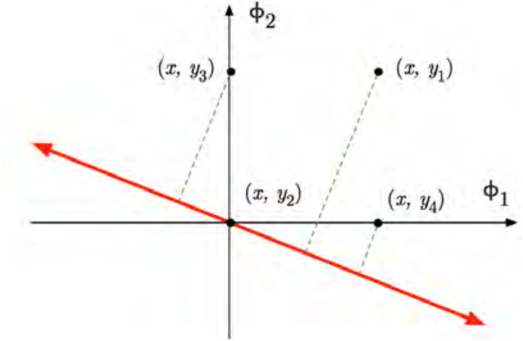
- Suppose we an instance $\boldsymbol{w}$ and $\mathcal{L} = \{y_1, y_2, y_3, y_4\}$.
- We have two simple binary features ϕ_1, and ϕ_2
- Suppose $\boldsymbol{w}$ is such that $\boldsymbol{w} \cdot \boldsymbol{\Phi} = w_1\phi 1 + w_2\phi_2$



$\mathbf{w} \cdot \boldsymbol{\phi} = w_1\phi_1 + w_2\phi_2 = 0$

A Geometric View of Linear Classifiers

- Suppose we an instance $\boldsymbol{w}$ and $\mathcal{L} = \{y_1, y_2, y_3, y_4\}$.
- We have two simple binary features ϕ_1, and ϕ_2
- Suppose $\boldsymbol{w}$ is such that $\boldsymbol{w} \cdot \boldsymbol{\Phi} = w_1 \phi 1 + w_2 \phi_2$

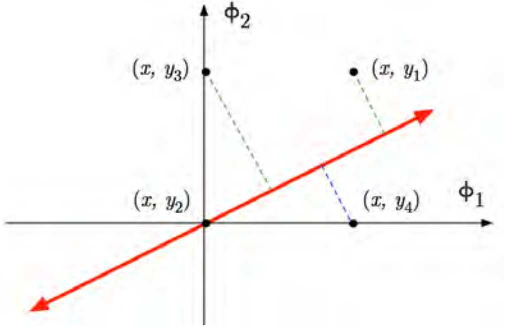


$$\text{distance}(\boldsymbol{w} \cdot \boldsymbol{\Phi}, \boldsymbol{\Phi}_0) = \frac{|\boldsymbol{w} \cdot \boldsymbol{\Phi}_0|}{\|\boldsymbol{w}\|_2} \propto |\boldsymbol{w} \cdot \boldsymbol{\Phi}_0|$$

- So $\boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, y_1) > \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, y_3) > \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, y_4) > \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, y_2) \geq 0$

A Geometric View of Linear Classifiers

- Suppose we an instance $\boldsymbol{w}$ and $\mathcal{L} = \{y_1, y_2, y_3, y_4\}$.
- We have two simple binary features ϕ_1, and ϕ_2
- Suppose $\boldsymbol{w}$ is such that $\boldsymbol{w} \cdot \boldsymbol{\Phi} = w_1 \phi 1 + w_2 \phi_2$



- So $\boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, y_3) > \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, y_1) > \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, y_2) > \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, y_4)$

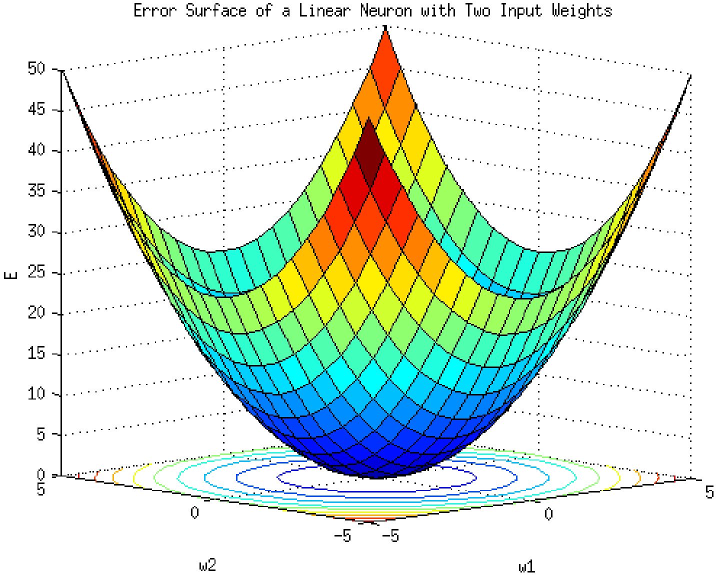
Where do we get $\boldsymbol{w}$? The Perceptron Learner

- Start with $\boldsymbol{w} = \mathbf{0}$
- Go over the training samples and adjust $\boldsymbol{w}$ to minimize the deviation from correct labels.

$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} \left(\max_{\ell' \in \mathcal{L}} \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_i, \ell') \right) - \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_i, \ell_i)$$

- The *perceptron learning algorithm* is a stochastic subgradient descent algorithm on above.
- For $t \in \{1, \ldots, T\}$
  - Pick i_t uniformly at random from $\{1, \ldots, n\}$
  - $\hat{\ell}_{i_t} \leftarrow \arg\max_{\ell \in \mathcal{L}} \; \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_{i_t}, \ell)$
  - $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \left(\boldsymbol{\Phi}(\boldsymbol{x}_{i_t}, \hat{\ell}_{i_t}) - \boldsymbol{\Phi}(\boldsymbol{x}_{i_t}, \ell_{i_t}) \right)$
- Return $\boldsymbol{w}$

Gradient Descent



More Sophisticated Classification

- Take into account *error costs* if all mistakes are not equally bad. (false positives vs. false negatives in spam detection)
- Use *maximum margin techniques* (e.g., Support Vector Machines) try to find the best separating hyperplane that's far from the training examples.
- Use *kernel methods* map vectors to get much higher-dimensional spaces, almost for free, where they may be lineraly separable.
- Use *Feature selection* to find the most important features and throw out the rest.
- Take the machine learning class if you are interested on these

11-411
Natural Language Processing
Part-of-Speech Tagging

Kemal Oflazer

Carnegie Mellon University in Qatar

Motivation

- My cat, which **lives** dangerously, no longer has nine **lives**.
  - The first **lives** is a present tense verb.
  - The second **lives** is a plural noun.
  - They are pronounced differently.
- How we pronounce the word depends on us knowing which is which.
  - The two **lives** above are pronounced differently.
  - “The **minute** issue took one **minute** to resolve.”
  - They can be stressed differently. **SUSpect** (noun) vs. **susPECT** (verb)
- He **can can** the **can**.
  - The first **can** is a modal.
  - The second **can** is a(n untensed) verb.
  - The third **can** is a singular noun.
  - In fact, **can** has one more possible interpretation as a present tense verb as in “We can tomatotes every summer.”

What are Part-of-Speech Tags?

- A limited number of tags to denote words “classes”.
- Words in the same class
  - Occur more or less in the same contexts
  - Have more or less the same functions
  - Morphologically, they (usually) take the same suffixes or prefixes.
- Part-of-Speech tags are not about meaning!
- Part-of-Speech tags are not necessarily about any grammatical function.

English Nouns

- Can be subjects and objects
  - This book is about geography.
  - I read a good book.
- Can be plural or singular (books, book)
- Can have determiners (the book)
- Can be modified by adjectives (blue book)
- Can have possessors (my book, John’s book)

Why have Part-of-Speech Tags?

- It is an “abstraction” mechanism.
- There are too many words.
  - You would need a lot of data to train models.
  - Your model would be very specific.
- POS Tags allow for generalization and allow for useful reduction in model sizes.
- There are many different tagsets: You want the right one for your task

How do we know the class?

- Substitution test
  - The ADJ cat sat on the mat.
  - The blue NOUN sits on the NOUN.
  - The blue cat VERB on the mat.
  - The blue cat sat PP the mat.

What are the Classes?

- Nouns, Verbs, Adjectives, . . .
  - Lots of different values (open class)
- Determiners
  - The, a, this, that, some, . . .
- Prepositions
  - By, at, from, as, against, below, . . .
- Conjunctions
  - And, or, neither, but, . . .
- Modals
  - Will, may, could, can, . . .
- Some classes are well defined and *closed*, some are *open*.

Broad Classes

- **Open Classes**: nouns, verbs, adjectives, adverbs, numbers
- **Closed Classes**: prepositions, determiners, pronouns, conjunctions, auxiliary verbs, particles, punctuation

Finer-grained Classes

- **Nouns**: Singular, Plural, Proper, Count, Mass
- **Verbs**: Untensed, Present 3rd Person, Present Non-3rd Person, Past , Past Participle
- **Adjectives**: Normal, Comparative, Superlative
- **Adverbs**: Comparative, Superlative, Directional, Temporal, Manner,
- **Numbers**: Cardinal, Ordinal

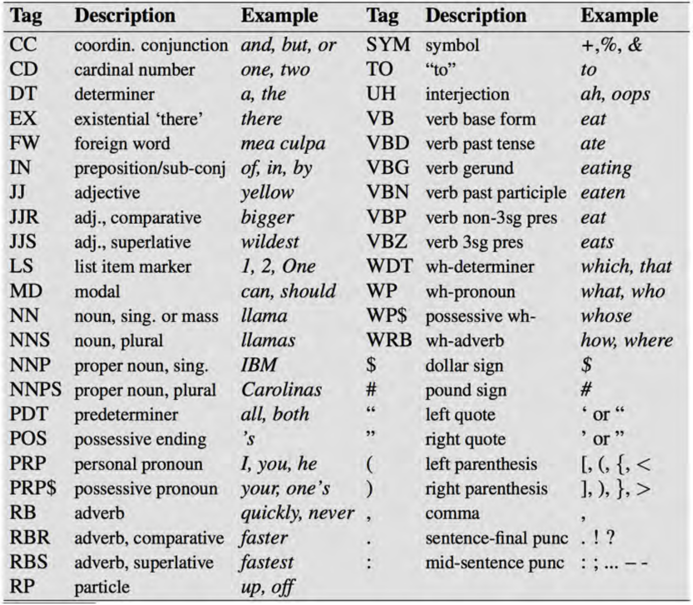
Hard Cases

- I will call up my friend.
- I will call my friend up.
- I will call my friend up in the treehouse.
- Gerunds
  - I like walking.
  - I like apples.
  - His walking daily kept him fit.
  - His apples kept him fit.
  - Eating apples kept him fit.

Other Classes

- Interjections (Wow!, Oops, Hey)
- Politeness markers (Your Highness . . .)
- Greetings (Dear . . .
- Existential there (there is . . .)
- Symbols, Money, Emoticons, URLs, Hashtags

Penn Treebank Tagset for English



Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	“to”	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential ‘there’	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	$
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	“	left quote	‘ or “
POS	possessive ending	*’s*	”	right quote	’ or ”
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one’s*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Others Tagsets for English and for Other Languages

- The International Corpus of English (ICE) Tagset: 205 Tags
- London-Lund Corpus (LLC) Tagset: 211 Tags
- Arabic: Several tens of (composite tags)
  - وسيكتبونها (Buckwalter: `wsyktbwnhA` “And they will write it”) is tagged as `CONJ + FUTURE PARTICLE + IMPERFECT VERB PREFIX + IMPERFECT VERB + IMPERFECT VERB SUFFIX MASCULINE PLURAL 3RD PERSON + OBJECT PRONOUN FEMININE SINGULAR`
- Czech: Several hundred (composite) tags
  - *Vaclav* is tagged as `k1gMnSc1`, indicating it is a noun, gender is male animate, number is singular, and case is nominative
- Turkish: Potentially infinite set of (composite) tags.
  - *elmasında* is tagged as `elma+Noun+A3sg+P3sg+Loc` indicating root is *elma* and the word is singular noun belonging to a third singular person in locative case.

Some Tagged Text from The Penn Treebank Corpus

```
In/IN an/DT Oct./NNP 19/CD review/NN of/IN ``/`` The/DT Misanthrope/NN
''/'' at/IN Chicago/NNP 's/POS Goodman/NNP Theatre/NNP  ``/``
Revitalized/VBN Classics/NNS Take/VBP the/DT Stage/NN in/IN Windy/NNP
City/NNP ,/, ''/'' Leisure/NN &/CC Arts/NNS ,/, the/DT role/NN of/IN
Celimene/NNP ,/, played/VBN by/IN Kim/NNP Cattrall/NNP , , was/VBD
mistakenly/RB attributed/VBN to/TO Christina/NNP Haag/NNP ./.
Ms./NNP Haag/NNP plays/VBZ Elianti/NNP ./.
Rolls-Royce/NNP Motor/NNP Cars/NNPS Inc./NNP said/VBD it/PRP
expects/VBZ its/PRP$ U.S./NNP sales/NNS to/TO remain/VB steady/JJ
at/IN about/IN 1,200/CD cars/NNS in/IN 1990/CD ./.
The/DT luxury/NN auto/NN maker/NN last/JJ year/NN sold/VBD 1,214/CD
cars/NNS in/IN the/DT U.S./NNP
```

How Bad is Ambiguity?

Tags	Token
7	down
6	that
6	set
6	put
6	open
6	hurt
6	cut
6	bet
6	back
5	vs,
5	the
5	spread
5	split
5	say
5	’s

Tags	Token
5	run
5	repurchase
5	read
5	present
5	out
5	many
5	less
5	left

Count	POS/Token
317	RB/down
200	RP/down
138	IN/down
10	JJ/down
1	VBP/down
1	RBR/down
1	NN/down

Some Tags for “down”

```
One/CD hundred/CD and/CC ninety/CD two/CD former/JJ greats/NNS ,/, near/JJ
greats/NNS ,/, hardly/RB knowns/NNS and/CC unknowns/NNS begin/VBP a/DT 72-game/JJ
,/, three-month/JJ season/NN in/IN spring-training/NN stadiums/NNS up/RB and/CC
down/RB Florida/NNP ...
He/PRP will/MD keep/VB the/DT ball/NN down/RP ,/, move/VB it/PRP around/RB ...
As/IN the/DT judge/NN marched/VBD down/IN the/DT center/JJ aisle/NN in/IN his/PRP$
flowing/VBG black/JJ robe/NN ,/, he/PRP was/VBD heralded/VBN by/IN a/DT trumpet/NN
fanfare/NN ...
Other/JJ Senators/NNP want/VBP to/TO lower/VB the/DT down/JJ payments/NNS
required/VBN on/IN FHA-insured/JJ loans/NNS ...
Texas/NNP Instruments/NNP ,/, which/WDT had/VBD reported/VBN Friday/NNP that/IN
third-quarter/JJ earnings/NNS fell/VBD more/RBR than/IN 30/CD %/NN from/IN the/DT
year-ago/JJ level/NN ,/, went/VBD down/RBR 2/CD 1/8/CD to/TO 33/CD on/IN 1.1/CD
million/CD shares/NNS ....
Because/IN hurricanes/NNS can/MD change/VB course/NN rapidly/RB ,/, the/DT
company/NN sends/VBZ employees/NNS home/NN and/CC shuts/NNS down/VBP
operations/NNS in/IN stages/NNS : /:  the/DT closer/RBR a/DT storm/NN gets/VBZ ,/,
the/DT more/RBR complete/JJ the/DT shutdown/NN ...
Jaguar/NNP ’s/POS American/JJ depositary/NN receipts/NNS were/VBD up/IN 3/8/CS
yesterday/NN in/IN a/DT down/NN market/NN ,/, closing/VBG at/IN 10/CD ...
```

Some Tags for “Japanese

Meanwhile/RB ,/, Japanese/JJ bankers/NNS said/VBD they/PRP were/VBD still/RB hesitant/JJ about/IN accepting/VBG Citicorp/NNP ’s/POS latest/JJS proposal/NN ...

And/CC the/DT Japanese/NNPS are/VBP likely/JJ to/TO keep/VB close/RB on/IN Conner/NNP ’s/POS heels/NNS ...

The/DT issue/NN is/VBZ further/RB complicated/VBN because/IN although/IN the/DT organizations/NNS represent/VBP Korean/JJ residents/NNS ,/, those/DT residents/NNS were/VBD largely/RB born/VBN and/CC raised/VBN in/IN Japan/NNP and/CC many/JJ speak/VBP only/RB Japanese/NNP ...

And/CC the/DT Japanese/NNP make/VBP far/RB more/JJR suggestions/NNS :/: 2,472/CS per/IN 100/CD eligible/JJ employees/NNS vs./CC only/RB 13/CD per/IN 100/CD employees/NNS in/IN the/DT ...

The/DT Japanese/NNS are/VBP in/IN the/DT early/JJ stage/NN right/RB now/RB ,/, said/VBD Thomas/NNP Kenney/NNP ,/, a/DT onetime/JJ media/NN adviser/NN for/IN First/NNP Boston/NNP Corp./NNP who/WP was/VBD recently/RB appointed/VBN president/NN of/IN Reader/NNP ’s/POS Digest/NNP Association/NNP ’s/POS new/JJ Magazine/NNP Publishing/NNP Group/NNP ...

In/IN 1991/CD ,/, the/DT Soviets/NNS will/MD take/VB a/DT Japanese/JJ into/NN space/NN ,/, the/DT first/JJ Japanese/NN to/TO go/VB into/IN orbit/NN ...

How we do POS Tagging?

- Pick the most frequent tag for each type
  - Gives about 92.34% accuracy (on a standard test set)
- Look at the context
  - Preceeding (and succeeding) words
  - Preceeding (and succeeding) tags
  - the . . .
  - to . . .
  - John’s blue . . .

Markov Models for POS Tagging

- We use an *already annotated* training data to statistically model POS tagging.
- Again the problem can be cast as a noisy channel problem:
  - "I have a sequence of tags of a proper sentence in my mind, $\boldsymbol{t} = \langle t_1, t_2, \ldots, t_n \rangle$"
  - "By the time, the tags are communicated, they are turned into actual words, $\boldsymbol{w} = \langle w_1, w_2, \ldots, w_n \rangle$, which are observed."
  - "What is the most likely tag sequence $\hat{\boldsymbol{t}}$ that gives rise to the observation $\boldsymbol{w}$? "
- The basic equation for tagging is then

$$\hat{\boldsymbol{t}} = \arg\max_{\boldsymbol{t}} \; p(\boldsymbol{t} \mid \boldsymbol{w})$$

where $\hat{\boldsymbol{t}}$ is the tag sequence that maximizes the argument of the $\arg\max$.

Basic Equation and Assumptions for POS Tagging

$$\hat{\boldsymbol{t}} = \arg\max_{\boldsymbol{t}} p(\boldsymbol{t} \mid \boldsymbol{w}) = \underbrace{\arg\max_{\boldsymbol{t}} \frac{p(\boldsymbol{w} \mid \boldsymbol{t})p(\boldsymbol{t})}{p(\boldsymbol{w})}}_{\text{Bayes Expansion}} = \underbrace{\arg\max_{\boldsymbol{t}} \overbrace{p(\boldsymbol{w} \mid \boldsymbol{t})}^{\text{Channel Model}} \overbrace{p(\boldsymbol{t})}^{\text{Source Model}}}_{\text{Ignoring Denominator}}$$

- The **independence** assumption: Probability of a word appearing depends only on its own tag and is independent of neighboring words and tags:

$$p(\boldsymbol{w} \mid \boldsymbol{t}) = p(w_{1:n} \mid t_{1:n}) \approx \prod_{i=1}^{n} p(w_i \mid t_i)$$

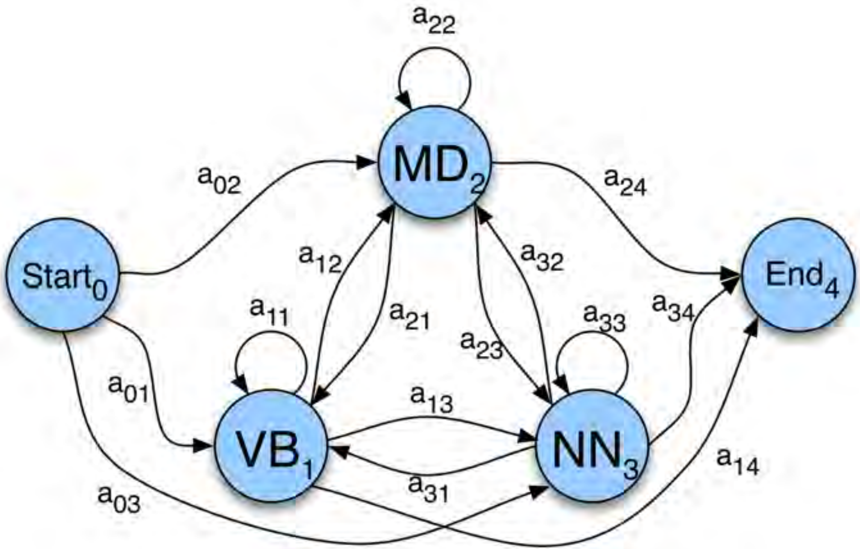
- The **bigram** assumption: that probability of a tag is dependent only on the previous tag.

$$p(\boldsymbol{t}) = p(t_{1:n}) \approx \prod_{i=1}^{n} p(t_i \mid t_{i-1})$$

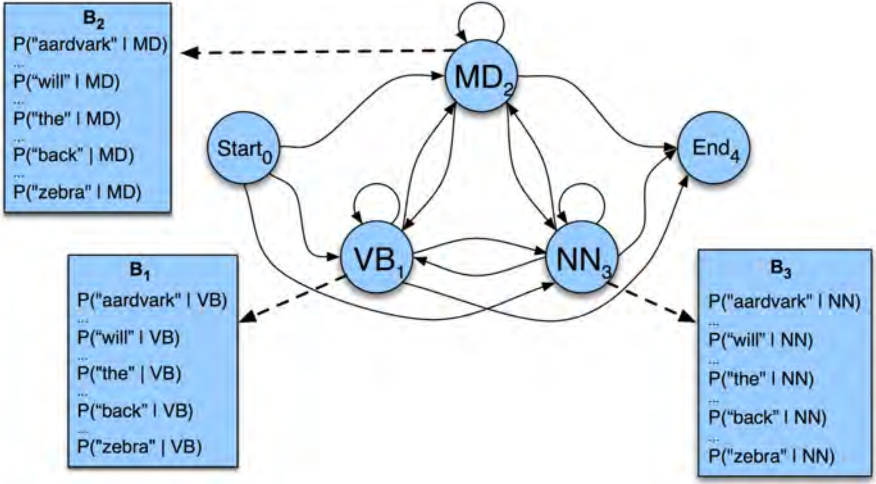
Basic Approximation Model for Tagging

$$\hat{t}_{1:n} = \arg\max_{t_{1:n}} \; p(t_{1:n} \mid w_{1:n}) \approx \arg\max_{t_{1:n}} \; \prod_{i=1}^{n} \underbrace{p(w_i \mid t_i)}_{emission} \underbrace{p(t_i \mid t_{t-1})}_{transition}$$

Bird's Eye View of $p(t_i \mid t_{i-1})$



Bird's Eye View of $p(w_i \mid t_i)$



Estimating Probabilities

- We can estimate these probabilities from a tagged training using maximum likelihood estimation.
- Transition Probabilities: $p(t_i \mid t_{i-1}) = \frac{c(t_{i-1}, t_i)}{c(t_{i-1})}$
- Emission Probabilities: $p(w_i \mid t_i) = \frac{c(t_i, w_i)}{c(t_i)}$

- It is also possible to use a trigram approximation (with appropriate smoothing).

$$p(t_{1:n}) \approx \prod_{i=1}^{n} p(t_i \mid t_{i-2} t_{i-1})$$

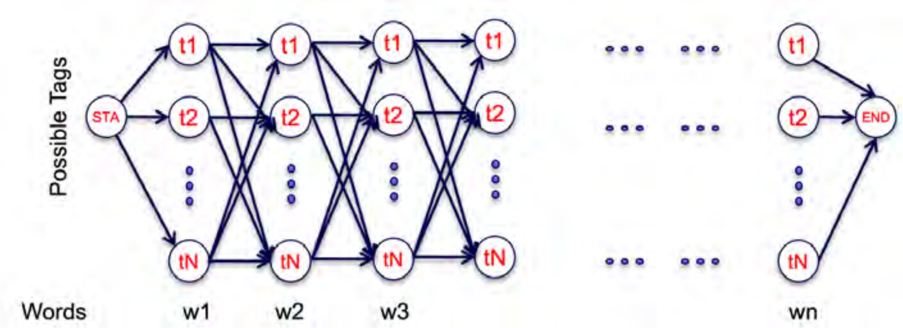
  - You need to square the number of states!

The Setting

- We have n words in $\boldsymbol{w} = \langle w_1 w_2, \ldots, w_n \rangle$.
- We have total N tags which are the labels of the Markov Model states (excluding *start* (0) and *end* (F) states).
- q_i is the label of the state after i words have been observed.
- We will also denote all the parameters of our HMM by $\lambda = (A, B)$, the transition (A) and emission (B) probabilities.
- In the next several slides
  - i will range over word positions.
  - j will range over states/tags
  - k will range over states/tags.

The Forward Algorithm

- An efficient dynamic programming algorithm for finding the total probability of observing $\boldsymbol{w} = \langle w_1, w_2, \ldots, w_n \rangle$, given the (Hidden) Markov Model λ.
- Creates expanded directed acyclic graph that is a specialized version of the model graph to the specific sentence called a *trellis*.



The Forward Algorithm

- Computes $\alpha_i(j) = p(w_1, w_2, \ldots, w_i, q_i = j \mid \lambda)$
  - The total probability of observing $w_1, w_2, \ldots, w_i$ and landing in state j after emitting i words.
- Let's define some short-cuts:
  - $\alpha_{i-1}(k)$: the previous forward probability from the previous stage (word)
  - $a_{kj} = p(t_j \mid t_k)$
  - $b_j(w_i) = p(w_i \mid t_j)$
- $\alpha_i(j) = \sum_{k=1}^{N} \alpha_{i-1}(k) \cdot a_{kj} \cdot b_j(w_i)$
- $\alpha_n(F) = p(w_1, w_2, \ldots, w_n, q_n = F \mid \lambda)$ is the total probability of observing $w_1, w_2, \ldots, w_n$.
- We really do not need αs. We just wanted to motivate the trellis.
- We are actually interested in the **most likely sequence of states (tags)** that we go through while "emitting" $w_1, w_2, \ldots, w_i$ These would be the most likely tags!.

Viterbi Decoding

- Computes $v_i(j) = \max_{q_0, q_1, \ldots q_{i-1}} p(q_0, q_1, \ldots q_{i-1}, w_1, w_2, \ldots, w_i, q_i = j \mid \lambda)$
- $v_i(j)$ is the maximum probability of observing $w_1, w_2, \ldots, w_i$ after emitting i words while going through some sequence of states (tags) $q_0, q_1, \ldots q_{i-1}$ before landing in state $q_i = j$.
- We can recursively define

$$v_i(j) = \max_{k=1 \ldots N} v_{i-1}(k) \cdot a_{kj} \cdot b_k(w_i)$$

- Let's also define a backtrace pointer as

$$bt_i(j) = \arg\max_{k=1 \ldots N} \; v_{i-1}(k) \cdot a_{kj} \cdot b_k(w_i)$$

- These backtrace pointers will give us the tag sequence $q_0 =$ START$, q_1, q_2, \ldots, q_n$ which is the most likely tag sequence for $\langle w_1, w_2, \ldots, w_n \rangle$.

Viterbi Algorithm

- **Initialization:**

$$\begin{aligned} v_1(j) &= a_{0j} \cdot b_j(w_1) \quad 1 \leq j \leq N \\ bt_1(j) &= 0 \end{aligned}$$

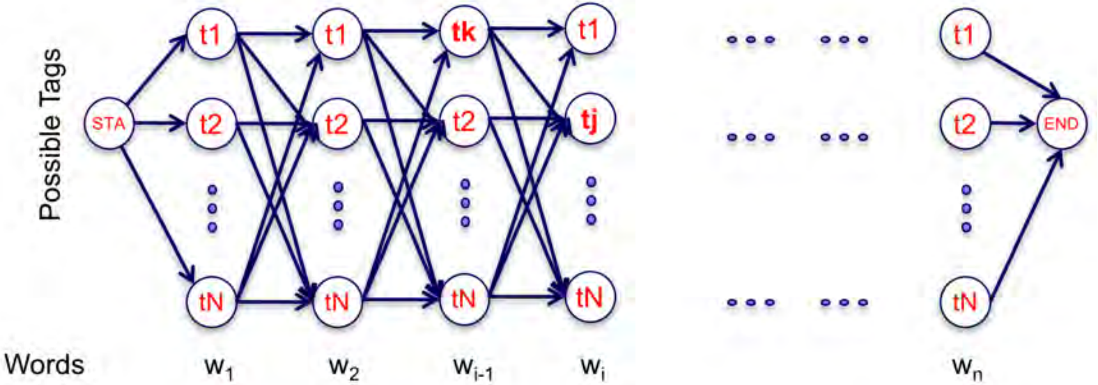
- **Recursion:**

$$\begin{aligned} v_i(j) &= \max_{k=1 \ldots N} v_{i-1}(k) \cdot a_{kj} \cdot b_j(w_i) \quad 1 \leq j \leq N, 1 < i \leq n \\ bt_1(j) &= \arg\max_{k=1 \ldots N} v_{i-1}(k) \cdot a_{kj} \cdot b_j(w_i) \quad 1 \leq j \leq N, 1 < i \leq n \end{aligned}$$

- **Termination:**

$$\begin{aligned} p* = v_n(q_F) &= \max_{k=1 \ldots N} v_n(k) \cdot a_{jF} \quad \text{The best score} \\ q_n* = bt_n(q_F) &= \arg\max_{k=1 \ldots N} v_n(k) \cdot a_{jF} \quad \text{The start of the backtrace} \end{aligned}$$

Viterbi Decoding

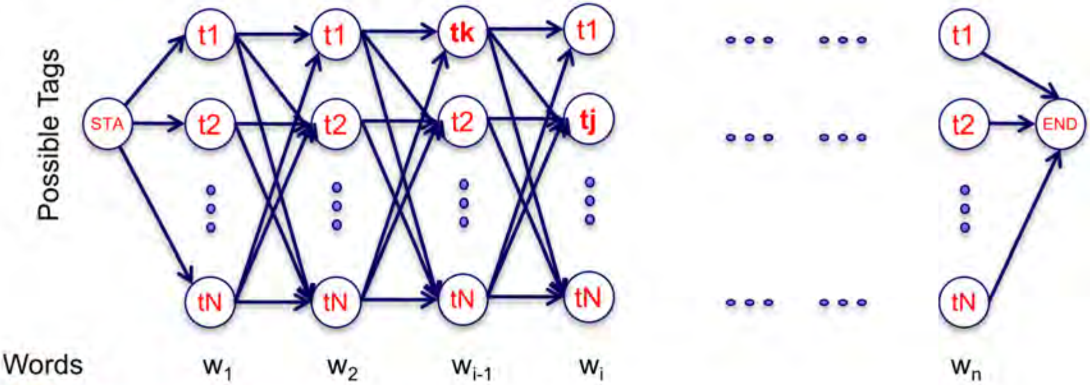


For all i

For all j, compute

$$v_i(t_j) = \max_k(v_{i-1}(t_k)\times p(w_i|t_j)\times p(t_j|t_k))$$

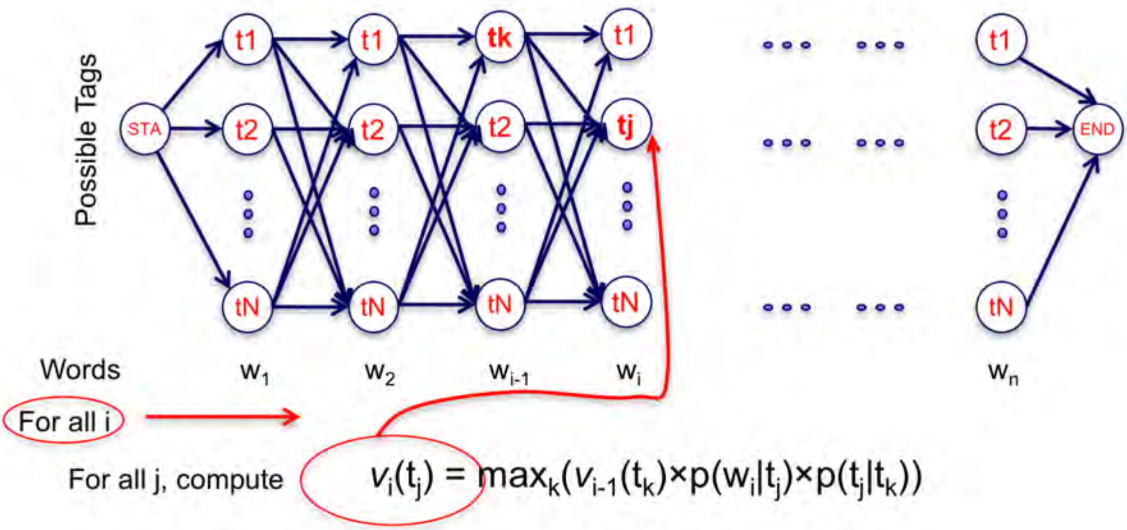
Viterbi Decoding



For all i

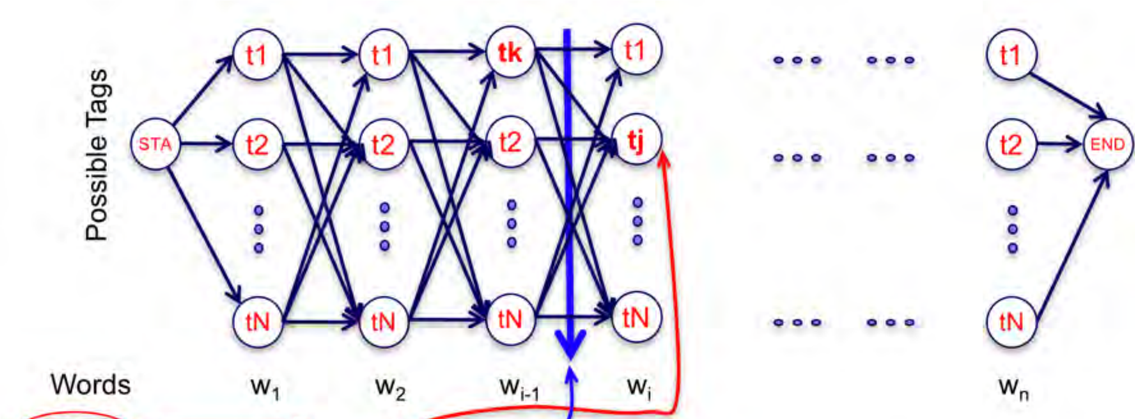
For all j, compute $v_i(t_j) = \max_k(v_{i-1}(t_k) \times p(w_i|t_j) \times p(t_j|t_k))$

Viterbi Decoding



For all j, compute $v_i(t_j) = \max_k(v_{i-1}(t_k)\times p(w_i|t_j)\times p(t_j|t_k))$

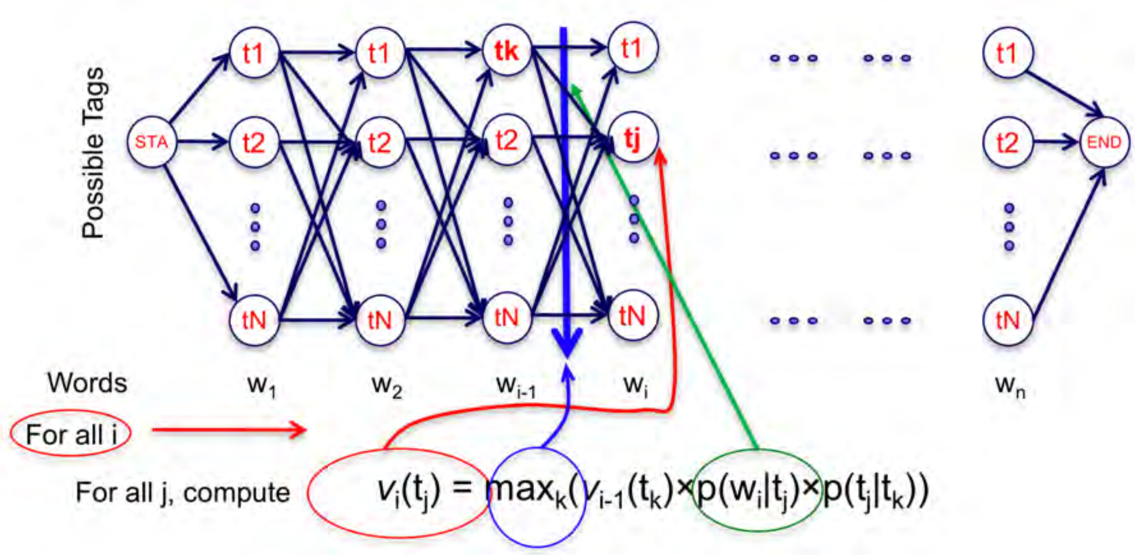
Viterbi Decoding



For all i

For all j, compute $v_i(t_j) = \max_k(v_{i-1}(t_k) \times p(w_i|t_j) \times p(t_j|t_k))$

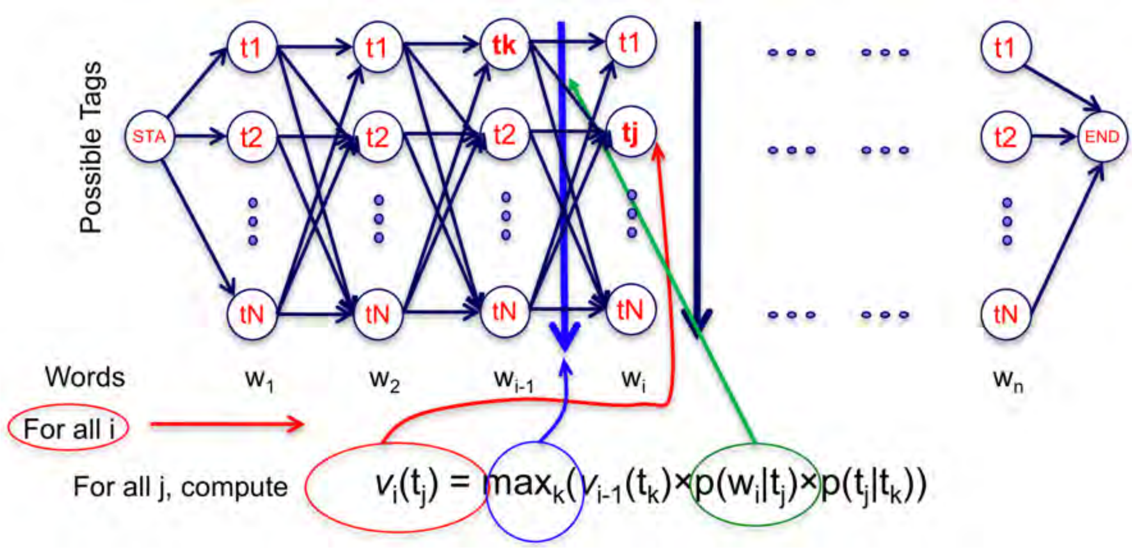
Viterbi Decoding



For all i

For all j, compute $v_i(t_j) = \max_k(v_{i-1}(t_k) \times p(w_i|t_j) \times p(t_j|t_k))$

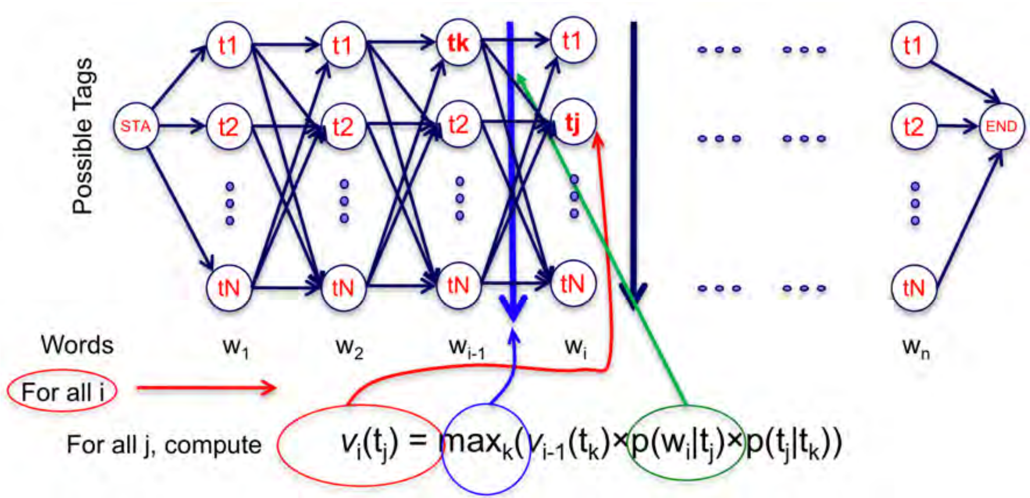
Viterbi Decoding



For all i

For all j, compute $v_i(t_j) = \max_k(v_{i-1}(t_k) \times p(w_i|t_j) \times p(t_j|t_k))$

Viterbi Decoding

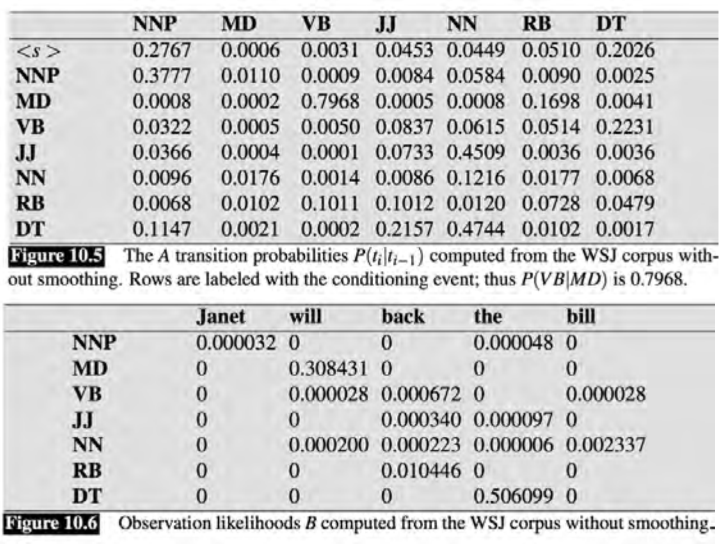


For all i

For all j, compute $v_i(t_j) = \max_k(v_{i-1}(t_k) \times p(w_i|t_j) \times p(t_j|t_k))$

- Once you are at $i = n$, you have to land in the *END* state (F), then use the backtrace to find the previous state you came from and recursively trace backwards to find $\hat{t}_{1:n}$.

Viterbi Decoding Example



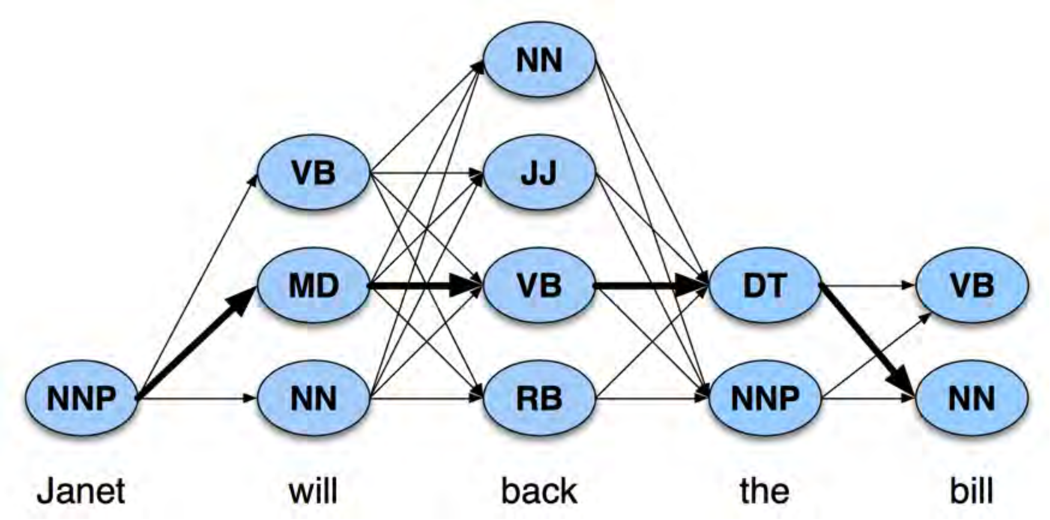
	NNP	MD	VB	JJ	NN	RB	DT
$<s>$	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Figure 10.5 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

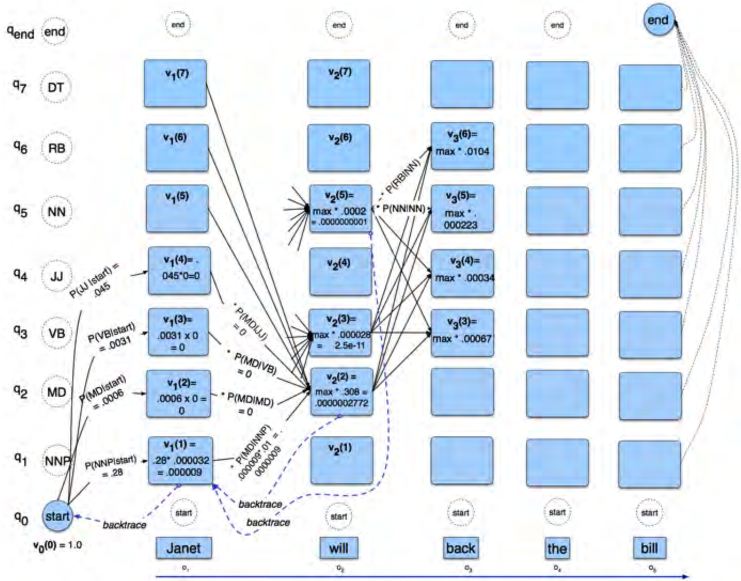
	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 10.6 Observation likelihoods B computed from the WSJ corpus without smoothing.

Viterbi Decoding Example



Viterbi Decoding Example



Unknown Words

- They are unlikely to be closed class words.
- They are most likely to be nouns or proper nouns, less likely, verbs.
- Exploit capitalization – most likely proper nouns.
- Exploit any morphological hints: *-ed* most likely past tense verb, *-s*, most likely plural noun or present tense verb for 3rd person singular.
- Build a separate models of the sort

  $$p(t_j \mid l_{n-i+1} \ldots l_n) \text{ and } p(l_{n-i+1} \ldots l_n)$$

  where $l_{n-i+1} \ldots l_n$ are the last i letters of a word.
- Then

  $$p(l_{n-i+1} \ldots l_n \mid t_j) = \frac{p(t_j \mid l_{n-i+1} \ldots l_n) \cdot p(l_{n-i+1} \ldots l_n)}{p(t_j)}$$

- Hence can be used in place of $p(w_i \mid t_i)$ in the Viterbi algorithm.
- Only use low frequency words in these models.

Closing Remarks

- Viterbi decoding takes $O(n \cdot N^2)$ work. (Why?)
- HMM parameters (transition probabilities A and emissions probabilities B) can actually be estimated from an unannotated corpus.
- Given an unannotated corpus and the state labels, the *forward-backward* or *Welch Welch algorithm*, a special case of the Expectation-Maximization (EM) algorithm trains both the transition probabilities A and the emission probabilities B of the HMM.
- EM is an iterative algorithm. It works by computing an initial estimate for the probabilities, then using those estimates to computing a better estimate, and so on, iteratively improving the probabilities that it learns.
- There are many other more recent and usually better performing approaches to POS tagging:
  - **Maximum Entropy Models** (discriminative, uses features, computes $\hat{\boldsymbol{t}} = \arg\max_{\boldsymbol{t}} p(\boldsymbol{t} \mid \boldsymbol{w})$)
  - **Conditional Random Fields** (discriminative, uses features, but features functions can also depend on the previous **tag**.)
  - **Perceptrons** (discriminative, uses features, trained with the perceptron algorithm)
- Accuracy for English is in the 97% to 98% range.
  - In every hundred words, you have 2 errors on the average and you do not know what they are!

11-411
Natural Language Processing
Overview of (Mostly English) Syntax

Kemal Oflazer

Carnegie Mellon University in Qatar

Syntax

- The ordering of words and how they group into phrases
  - [[the old man] [is yawning]]
  - [[the old] [man the boats]]
- Syntax vs. Meaning
  - “Colorless green ideas sleep furiously.”
  - You can tell that the words are in the right order.
  - and that “colorless” and “green” modify “ideas”
  - and that ideas sleep
  - that the sleeping is done furiously
  - that it sounds like an English sentence
  - if you can’t imagine what it means
  - and you know that it is better than “Sleep green furiously ideas colorless”

Syntax vs. Morphology

- Syntax is not morphology
  - Morphology deals with the internal structure of words.
  - Syntax deals with combinations of words – phrases and sentences.
- Syntax is mostly made up of general rules that apply across-the-board, with very little irregularities.

Syntax vs. Semantics

- Syntax is not semantics.
  - Semantics is about meaning; syntax is about structure alone.
  - A sentence can be syntactically well-formed but semantically ill-formed. (e.g., “Colorless green ideas sleep furiously.”)
- Some well-known linguistic theories attempt to “read” semantic representations off of syntactic representations in a compositional fashion.
  - We’ll talk about these in a later lecture

Two Approaches to Syntactic Structure

- **Constituent Structure** or **Phrase Structure Grammar**
  - Syntactic structure is represented by trees generated by a *context-free grammar*.
  - An important construct is the constituent (complete sub-tree).

- **Dependency Grammar**:
  - The basic unit of syntactic structure is a binary relation between words called a *dependency*.

Constituents

- One way of viewing the structure of a sentence is as a collection of nested constituents:
  - **Constituent**: a group of words that “go together” (or relate more closely to one another than to other words in the sentence)
- Constituents larger than a word are called *phrases*.
- Phrases can contain other phrases.

Constituents

- Linguists characterize constituents in a number of ways, including:
  - where they occur (e.g., “NPs can occur before verbs”)
  - where they can move in variations of a sentence
    - On September 17th, I’d like to fly from Atlanta to Denver.
    - I’d like to fly on September 17th from Atlanta to Denver.
    - I’d like to fly from Atlanta to Denver on September 17th.
  - what parts can move and what parts can’t
    - *On September I’d like to fly 17th from Atlanta to Denver.
  - what they can be conjoined with
    - I’d like to fly from Atlanta to Denver on September 17th and in the morning.

Noun Phrases

- The elephant arrived.
- It arrived.
- Elephants arrived.
- The big ugly elephant arrived.
- The elephant I love to hate arrived.

Prepositional Phrases

- Every prepositional phrase contains a preposition followed by a noun phrase.

- I arrived on Tuesday.
- I arrived in March.
- I arrived under the leaking roof.
- I arrived with the elephant I love to hate.

Sentences/Clauses

- John likes Mary.
- John likes the woman he thinks is Mary.
  - John likes the woman (whom) he thinks (the woman) is Mary.
- Sometimes, John thinks he is Mary.
  - Sometimes, John thinks (that) he/John is Mary.
- It is absolutely false that sometimes John thinks he is Mary.

Recursion and Constituents,

- This is the house.
- This is the house that Jack built.
- This is the cat that lives in the house that Jack built.
- This is the dog that chased the cat that lives in the house that Jack built.
- This is the flea that bit the dog that chased the cat that lives in the house the Jack built.
- This is the virus that infected the flea that bit the dog that chased the cat that lives in the house that Jack built.

- Non-constituents
  - If on a Winter's Night a Traveler
  - Nuclear and Radiochemistry
  - The Fire Next Time
  - A Tad Overweight, but Violet Eyes to Die For
  - Sometimes a Great Notion
  - [how can we know the] Dancer from the Dance

Describing Phrase Structure / Constituency Grammars

- Regular expressions were a convenient formalism for describing morphological structure of words.
- Context-free grammars are a convenient formalism for describing context-free languages.
- Context-free languages are a reasonable approximation for natural languages, while regular languages are much less so!
  - Although these depend on what the goal is.
- There is some linguistic evidence that natural languages are NOT context-free, but in fact are mildly context-sensitive.
  - This has not been a serious impediment.
- Other formalisms have been constructed over the years to deal with natural languages.
  - Unification-based grammars
  - Tree-adjoining grammars
  - Categorial grammars

11-411
Natural Language Processing
Formal Languages and Chomsky Hierarchy

Kemal Oflazer

Carnegie Mellon University in Qatar

Brief Overview of Formal Language Concepts

Strings

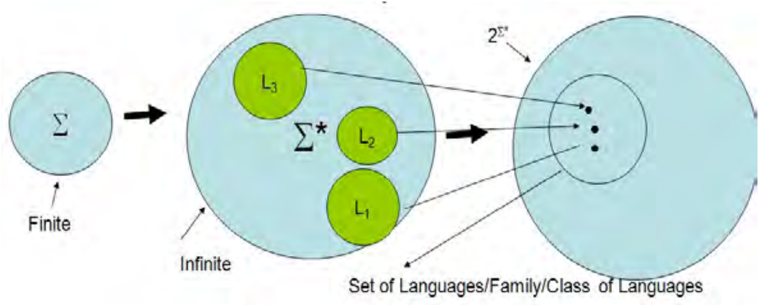
- An alphabet is any finite set of distinct symbols
  - {0, 1}, {0,1,2,. . . ,9}, {a,b,c}
  - We denote a generic alphabet by Σ
- A string is any finite-length sequence of elements of Σ.
- e.g., if $\Sigma = \{a, b\}$ then *a*, *aba*, *aaaa*, ..., *abababbaab* are some strings over the alphabet Σ

Strings

- The set of all possible strings over Σ is denoted by Σ^*.
- We define $\Sigma^0 = \{\epsilon\}$ and $\Sigma^n = \Sigma^{n-1} \cdot \Sigma$
  - with some abuse of the concatenation notation applying to sets of strings now
- So $\Sigma^n = \{\omega | \omega = xy \text{ and } x \in \Sigma^{n-1} \text{ and } y \in \Sigma\}$
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots \Sigma^n \cup \cdots = \bigcup_{0}^{\infty} \Sigma^i$
  - Alternatively, $\Sigma^* = \{x_1, \ldots, x_n | n \geq 0 \text{ and } x_i \in \Sigma \text{ for all } i\}$
- Φ denotes the empty set of strings $\Phi = \{\}$,
  - but $\Phi^* = \{\epsilon\}$

Sets of Languages

- The power set of Σ^*, the set of all its subsets, is denoted as 2^{Σ^*}



Describing Languages

- Interesting languages are infinite
- We need finite descriptions of infinite sets
  - $L = \{a^n b^n : n \geq 0\}$ is fine but not terribly useful!
- We need to be able to use these descriptions in mechanizable procedures

Describing Languages

- Regular Expressions/Finite State Recognizers $\Rightarrow$ Regular Languages
- *Context-free Grammars*/*Push-down Automata* $\Rightarrow$ Context-free Languages

Identifying Nonregular Languages

- Given language L how can we check if it is not a regular language ?
  - The answer is not obvious.
  - Not being able to design a DFA does not constitute a proof!

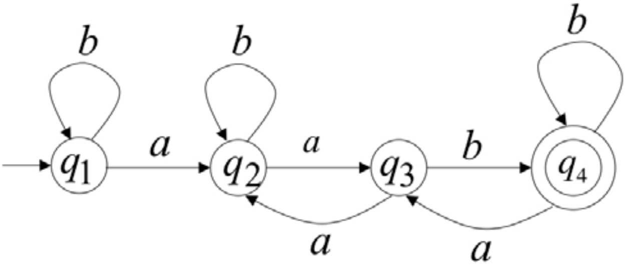
The Pigeonhole Principle

- If there are n pigeons and m holes and $n > m$, then at least one hole has > 1 pigeons.

- What do pigeons have to do with regular languages?

The Pigeonhole Principle

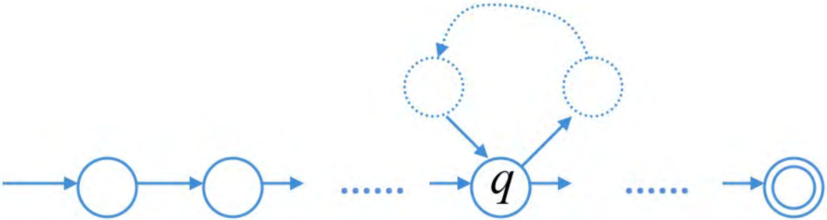
- Consider the DFA



- With strings a, aa or aab, no state is repeated
- With strings $aabb$, $bbaa$, $abbabb$ or $abbbabbabb$, a state is repeated
- In fact, for any ω where $|\omega| \geq 4$, some state has to repeat? Why?

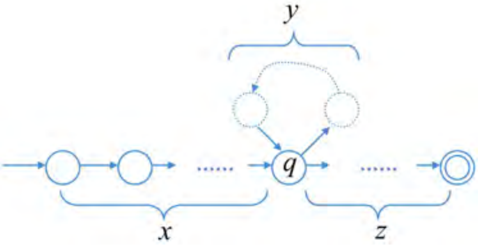
The Pigeonhole Principle

- When traversing the DFA with the string ω, if the number of transitions $\geq$ number of states, some state q has to repeat!
- Transitions are pigeons, states are holes.



Pumping a String

- Consider a string $\omega = xyz$



- $|y| \geq 1$
- $|xy| \leq m$ (m the number of states)
- If $\omega = xyz \in L$ that so are xy^iz for all $i \geq 0$
- The substring y can be pumped.
- So if a DFA accepts a sufficiently long string, then it accepts an infinite number of strings!

There are Nonregular Languages

- Consider the language $L = \{a^n b^n | n \geq 0\}$
- Suppose L is regular and a DFA with p states accepts L
- Consider $\delta^*(q_0, a^i)$ for $i = 0, 1, 2, \ldots$
  - $\delta^*(q, w)$ is the extended state transition function: what state do I land in starting in state q and stepping through the stmbols in w.
- Since there are infinite i's, but a finite number states, the Pigeonhole Principle tells us that there is some state q such that
  - $\delta^*(q_0, a^n) = q$ and $\delta^*(q_0, a^m) = q$, but $n \neq m$
  - Thus if M accepts $a^n b^n$ it must also accept $a^m b^n$, since in state q is does not "remember" if there were n or m a's.
- Thus M can not exist and L is not regular.

Is English Regular?

- The cat likes tuna fish.
- The cat the dog chased likes tuna fish.
- The cat the dog the rat bit chased likes tuna fish.
- The cat the dog the rat the elephant admired bit chased likes tuna fish.
- $L_1 =$ (the cat | the dog | the mouse| $\ldots$)* (chased | bit | ate | $\ldots$.)* likes tuna fish
- $L_2 =$ English
- $L_1 \cap L_2 =$ (the cat | the dog | the mouse| $\ldots$)^n (chased | bit | ate | $\ldots$.)$^{n-1}$ likes tuna fish.
- Closure fact: If L_1 and L_2 are regular $\Rightarrow L_1 \cap L_2$ is regular.
- L_1 is regular, $L_1 \cap L_2$ is NOT regular, hence L_2 (English) can NOT be regular.

Grammars

- Grammars provide the generative mechanism to generate all strings in a language.
- A grammar is essentially a collection of substitution rules, called productions
- Each production rule has a left-hand-side and a right-hand-side.

Grammars - An Example

- Consider once again $L = \{a^n b^n \mid n \geq 0\}$
- Basis: ϵ is in the language
  - Production: $S \rightarrow \epsilon$
- Recursion: If w is in the language, then so is the string awb.
  - Production: $S \rightarrow aSb$
- S is called a variable or a nonterminal symbol
- a, b etc., are called terminal symbols
- One variable is designated as the start variable or start symbol.

How does a grammar work?

- Consider the set of rules $R = \{S \to \epsilon, S \to aSb\}$
- Start with the start variable S
- Apply the following until all remaining symbols are terminal.
  - Choose a production in R whose left-hand sides matches one of the variables.
  - Replace the variable with the rule's right hand side.
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaaSbbbb \Rightarrow aaaabbbb$
- The string $aaaabbbb$ is in the language L
- The sequence of rule applications above is called a derivation.

Types of Grammars

- **Regular Grammars** describe regular languages.
- **Context-free Grammars**: describe context-free languages.
- **Context-sensitive Grammars**: describe context-sensitive languages.
- **General Grammars**: describe arbitrary Turing-recognizable languages.

Formal Definition of a Grammar

- A Grammar is a 4-tuple $G = (\mathcal{V}, \Sigma, R, S)$ where
  - $\mathcal{V}$ is a finite set of variables
  - Σ is a finite set of terminals, disjoint from $\mathcal{V}$.
  - R is a set of rules of the $X \to Y$
  - $S \in \mathcal{V}$ is the start variable
- In general $X \in (\mathcal{V} \cup \Sigma)^+$ and $Y \in (\mathcal{V} \cup \Sigma)^*$
- The type of a grammar (and hence the class of the languages described) depends on the type of the left- and right-hand sides.
- The right hand side of the rules can be any combination of variables and terminals, including ϵ (hence $Y \in (\mathcal{V} \cup \Sigma)^*$).

Types of Grammars

- Regular Grammars
  - Left-linear: All rules are either like $X \rightarrow Ya$ or like $X \rightarrow a$ with $X, Y \in \mathcal{V}$ and $a \in \Sigma^*$
  - Right-linear: All rules are either like $X \rightarrow aY$ or like $X \rightarrow a$ with $X, Y \in \mathcal{V}$ and $a \in \Sigma^*$
- Context-free Grammars
  - All rules are like $X \rightarrow Y$ with $X \in \mathcal{V}$ and $Y \in (\Sigma \cup \mathcal{V})^*$
- Context-sensitive Grammars
  - All rules are like $LXR \rightarrow Y$ with $X \in \mathcal{V}$ and $R, Y, L \in (\Sigma \cup \mathcal{V})^*$
- General Grammars
  - All rules are like $X \rightarrow Y$ with $X, Y \in (\Sigma \cup \mathcal{V})^*$

Chomsky Normal Form

- CFGs in certain standard forms are quite useful for some computational problems.

Chomsky Normal Form

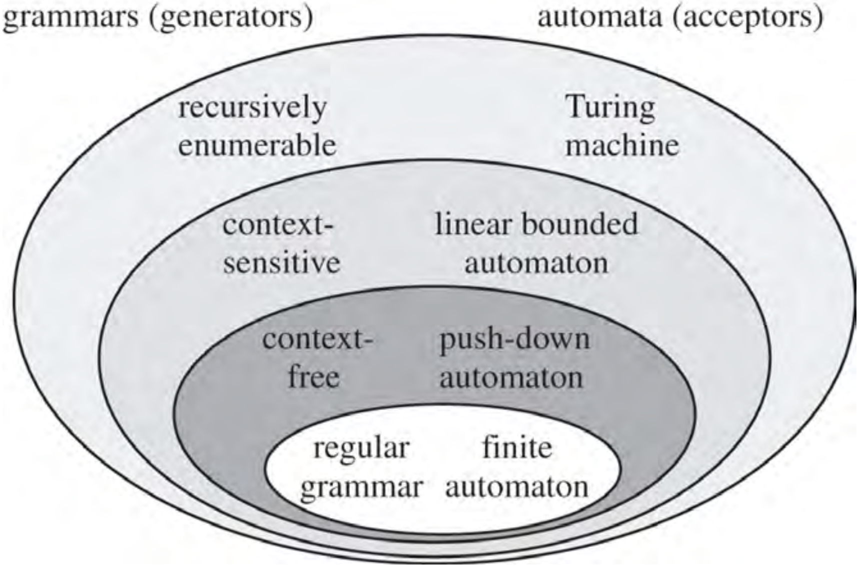
A context-free grammar is in Chomsky normal form(CNF) if every rule is either of the form

$$A \rightarrow BC \text{ or } A \rightarrow a$$

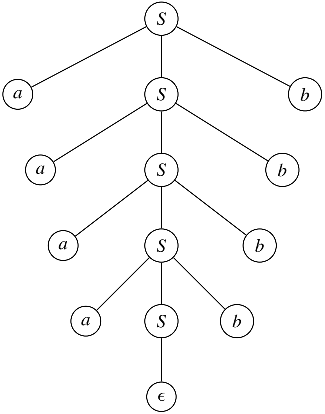
where a is a terminal and A, B, C are variables – except B and C may not be the start variable. In addition, we allow the rule $S \rightarrow \epsilon$ if necessary.

- Any CFG can be converted to a CFG in Chomsky Normal Form. They accept the same language but assign possibly different tree structures to the same string.

Chomsky Hierarchy



Parse Trees



The terminals concatenated from left to right give us the string.

- Derivations can also be represented with a parse tree.
- The leaves constitute the yield of the tree.
- Terminal symbols can occur only at the leaves.
- Variables can occur only at the internal nodes.

A Grammar for a Fragment of English

$$
\begin{array}{rcl}
S & \rightarrow & NP\ VP \\
NP & \rightarrow & CN \mid CN\ PP \\
VP & \rightarrow & CV \mid CV\ PP \\
PP & \rightarrow & P\ NP \\
CN & \rightarrow & DT\ N \\
CV & \rightarrow & V \mid V\ NP \\
DT & \rightarrow & \text{a} \mid \text{the} \\
N & \rightarrow & \text{boy} \mid \text{girl} \mid \text{flower} \mid \\
 & & \text{telescope} \\
V & \rightarrow & \text{touches} \mid \text{likes} \mid \\
 & & \text{sees} \mid \text{gives} \\
P & \rightarrow & \text{with} \mid \text{to}
\end{array}
$$

Nomenclature:

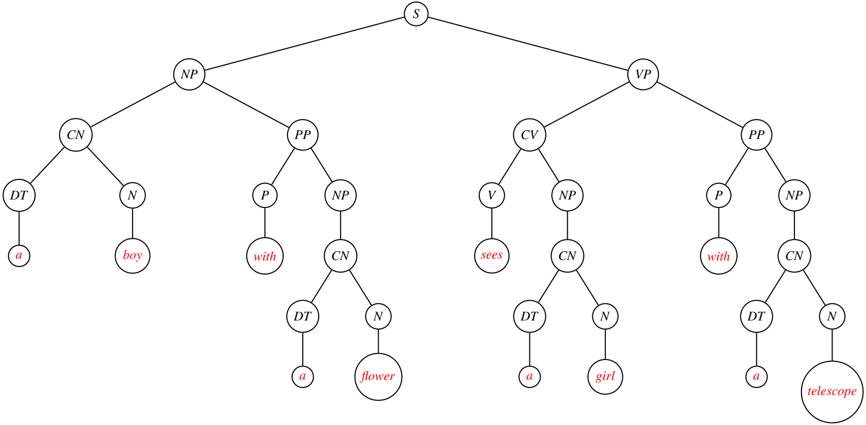
- *S*: Sentence
- *NP*: Noun Phrase
- *CN*: Complex Noun
- *PP*: Prepositional Phrase
- *VP*: Verb Phrase
- *CV*: Complex Verb
- *P*: Preposition
- *DT*: Determiner
- *N*: Noun
- *V*: Verb

A Grammar for a Fragment of English

$$\begin{array}{rcl}
S & \rightarrow & NP\ VP \\
NP & \rightarrow & CN \mid CN\ PP \\
VP & \rightarrow & CV \mid CV\ PP \\
PP & \rightarrow & P\ NP \\
CN & \rightarrow & DT\ N \\
CV & \rightarrow & V \mid V\ NP \\
DT & \rightarrow & \text{a} \mid \text{the} \\
N & \rightarrow & \text{boy} \mid \text{girl} \mid \text{flower} \mid \\
 & & \text{telescope} \\
V & \rightarrow & \text{touches} \mid \text{likes} \mid \\
 & & \text{sees} \mid \text{gives} \\
P & \rightarrow & \text{with} \mid \text{to}
\end{array}$$

$$\begin{array}{rcl}
S & \Rightarrow & NP\ VP \\
 & \Rightarrow & \underline{CN\ PP}\ VP \\
 & \Rightarrow & \underline{DT\ N}\ PP\ VP \\
 & \Rightarrow & \underline{\text{a}}\ N\ PP\ VP \\
 & \Rightarrow & \cdots \\
 & \Rightarrow & \text{a boy with a flower } VP \\
 & \Rightarrow & \text{a boy with a flower } \underline{CV\ PP} \\
 & \Rightarrow & \cdots \\
 & \Rightarrow & \text{a boy with a flower sees a girl} \\
 & & \text{with a telescope}
\end{array}$$

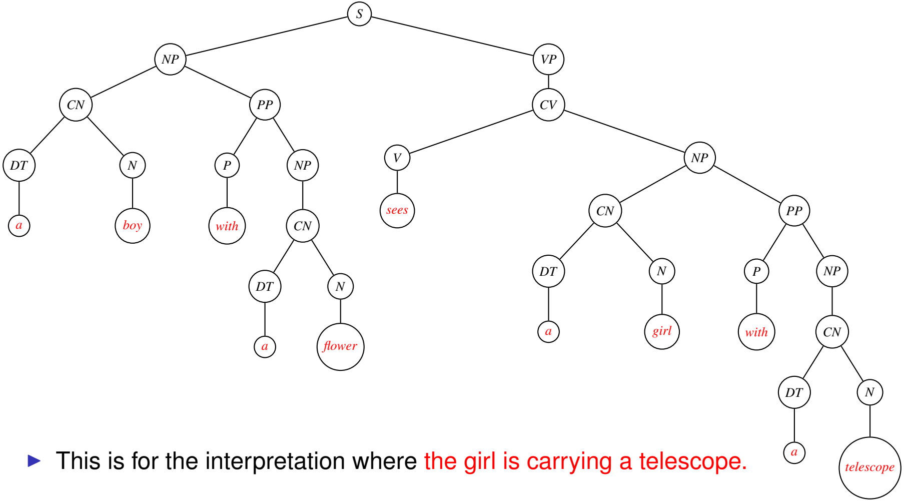
English Parse Tree



- This structure is for the interpretation where the boy is seeing with the telescope!

English Parse Tree

Alternate Structure



- This is for the interpretation where the girl is carrying a telescope.

Structural Ambiguity

- A set of rules can assign multiple structures to the same string.
- Which rule one chooses determines the eventual structure.
  - $VP \rightarrow CV \mid CV\ PP$
  - $CV \rightarrow V \mid V\ NP$
  - $NP \rightarrow CN \mid CN\ PP$
  - $\cdots$ $[_{VP}$ $[_{CV}$ sees $[_{NP}$ a girl] $[_{PP}$ with a telescope]].
  - $\cdots$ $[_{VP}$ $[_{CV}$ sees] $[_{NP}$ $[_{CN}$ a girl] $[_{PP}$ with a telescope]].
    - (Not all brackets are shown!)

Some NLP Considerations - Linguistic Grammaticality

- We need to address a wide-range of grammaticality.
- I'll write the company.
- I'll write to the company.
- It needs to be washed.
- It needs washed.
- They met Friday to discuss it.
- They met on Friday to discuss it.

Some NLP Considerations – Getting it Right

- CFGs provide you with a tool set for creating grammars
  - Grammars that work well (for a given application)
  - Grammars that work poorly (for a given application)
- There is nothing about the theory of CFGs that tells you, a priori, what a “correct” grammar for a given application looks like
- A good grammar is generally one that:
  - Doesn’t over-generate very much (high precision)
    - A grammar *over-generates* when it accepts strings not in the language.
  - Doesn’t under-generate very much (high recall)
    - A grammar *under-generates* when it does not accept strings in the language.

Some NLP Considerations – Why are we Building Grammars?

- Consider:
  - Oswald shot Kennedy.
  - Oswald, who had visited Russia recently, shot Kennedy.
  - Oswald assassinated Kennedy
- Who shot Kennedy?
- Consider
  - Oswald shot Kennedy.
  - Kennedy was shot by Oswald.
  - Oswald was shot by Ruby.
- Who shot Oswald?
- Active/Passive
  - Oswald shot Kennedy.
  - Kennedy was shot by Oswald.
- Relative clauses
  - Oswald who shot Kennedy was shot by Ruby.
  - Kennedy whom Oswald shot didn't shoot anybody.

Language Myths: Subject

- Myth I: the subject is the first noun phrase in a sentence
- Myth II: the subject is the actor in a sentence
- Myth III: the subject is what the sentence is about
- All of these are often true, but none of them is always true, or tells you what a subject really is (or how to use it in NLP).

Subject and Object

- Syntactic (not semantic)
  - The batter hit the ball. [subject is semantic agent]
  - The ball was hit by the batter. [subject is semantic patient]
  - The ball was given a whack by the batter. [subject is semantic recipient]
  - George, the key, the wind opened the door.
- Subject $\neq$ topic (the most important information in the sentence)
  - I just married the most beautiful woman in the world.
  - Now beans, I like.
  - As for democracy, I think it's the best form of government.
- English subjects
  - agree with the verb
  - when pronouns, are in nominative case (I/she/he vs. me/her/him)
- English objects
  - when pronouns, in accusative case (me, her, him)
  - become subjects in passive sentences

Looking Forward

- CFGs may not be entirely adequate for capturing the syntax of natural languages
  - They are *almost* adequate.
  - They are computationally well-behaved (in that you can build relatively efficient parsers for them, etc.)
  - But they are not very convenient as a means for handcrafting a grammar.
  - *They are not probabilistic*. But we will add probabilities to them soon.

Parsing Context-free Languages

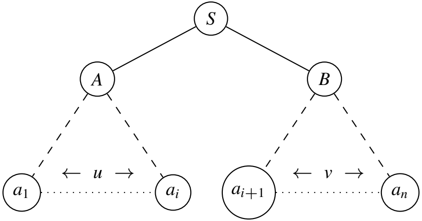
- The Cocke-Younger-Kasami (CYK) algorithm:
  - Grammar in Chomsky Normal Form (may not necessarily be linguistically meaningful)
  - All trees sanctioned by the grammar can be computed.
  - For an input of n words, requires $O(n^3)$ work (with a large constant factor dependent on the grammar size), using a bottom-up dynamic programming approach.
- Earley Algorithm:
  - Can handle arbitrary Context-free Grammars
  - Parsing is top-down.
  - Later.

The Cocke-Younger-Kasami (CYK) algorithm

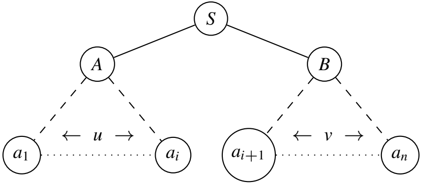
- The CYK parsing algorithm determines if $w \in L(G)$ for a grammar G in Chomsky Normal Form
  - with some extensions, it can also determine possible structures.
  - Assume $w \neq \epsilon$ (if so, check if the grammar has the rule $S \rightarrow \epsilon$)

The CYK Algorithm

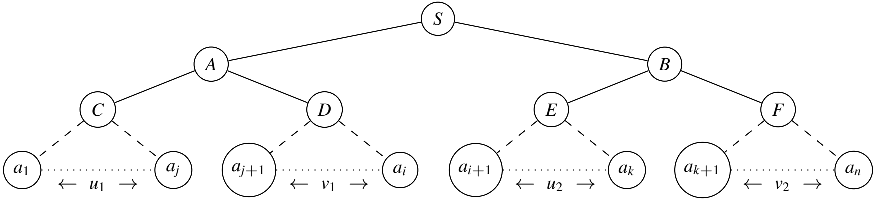
- Consider $w = a_1 a_2 \cdots a_n$, $a_i \in \Sigma$
- Suppose we could cut up the string into two parts $u = a_1 a_2 .. a_i$ and $v = a_{i+1} a_{i+2} \cdots a_n$
- Now suppose $A \overset{*}{\Rightarrow} u$ and $B \overset{*}{\Rightarrow} v$ and that $S \rightarrow AB$ is a rule.



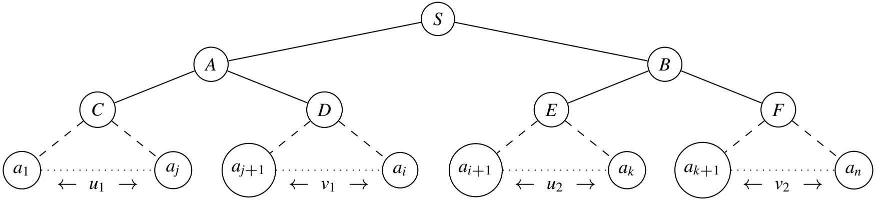
The CYK Algorithm



- Now we apply the same idea to A and B recursively.


S
A
B
C
D
E
F
a_1
a_j
$\leftarrow u_1 \rightarrow$
a_{j+1}
a_i
$\leftarrow v_1 \rightarrow$
a_{i+1}
a_k
$\leftarrow u_2 \rightarrow$
a_{k+1}
a_n
$\leftarrow v_2 \rightarrow$

The CYK Algorithm



S
A
B
C
D
E
F
a_1
a_j
a_{j+1}
a_i
a_{i+1}
a_k
a_{k+1}
a_n
$\leftarrow u_1 \rightarrow$
$\leftarrow v_1 \rightarrow$
$\leftarrow u_2 \rightarrow$
$\leftarrow v_2 \rightarrow$

- What is the problem here?
- We do not know what i, j and k are!
- No Problem! We can try all possible i's, j's and $k's$.
- Dynamic programming to the rescue.

DIGRESSION - Dynamic Programming

- An algorithmic paradigm
- Essentially like divide-and-conquer but subproblems overlap!
- Results of subproblem solutions are reusable.
- Subproblem results are computed once and then memoized
- Used in solutions to many problems
    - Length of longest common subsequence
    - Knapsack
    - Optimal matrix chain multiplication
    - Shortest paths in graphs with negative weights (Bellman-Ford Alg.)

(Back to) The CYK Algorithm

- Let $w = a_1 a_2 \cdots a_n$.
- We define
  - $w_{i,j} = a_i \cdots a_j$ (substring between positions i and j)
  - $V_{i,j} = \{A \in \mathcal{V} \mid A \overset{*}{\Rightarrow} w_{i,j}\}(j \geq i)$ (all variables which derive $w_{i,j}$)
- $w \in L(G)$ iff $S \in V_{1,n}$
- How do we compute $V_{i,j}(j \geq i)$?

The CYK Algorithm

- How do we compute $V_{i,j}$?
- Observe that $A \in V_{i,i}$ if $A \to a_i$ is a rule.
  - So $V_{i,\,i}$ can easily be computed for $1 \leq i \leq n$ by an inspection of w and the grammar.
- $A \overset{*}{\Rightarrow} w_{i,j}$ if
  - There is a production $A \to BC$, and
  - $B \overset{*}{\Rightarrow} w_{i,k}$ and $C \overset{*}{\Rightarrow} w_{k+1,j}$ for some k, $i \leq k < j$.
- So

$$V_{i,j} = \bigcup_{i \leq k < j} \{A :| \; A \to BC \text{ and } B \in V_{i,k} \text{ and } C \in V_{k+1,j}\}$$

The CYK Algorithm

$$V_{i,j} = \bigcup_{i \leq k < j} \{A : A \to BC \text{ and } B \in V_{i,k} \text{ and } C \in V_{k+1,j}\}$$

- Compute in the following order:

$$\begin{array}{cccccccc}
 & \rightarrow & & & & & & \\
\downarrow & V_{1,1} & V_{2,2} & V_{3,3} & \cdots & \cdots & \cdots & V_{n,n} \\
 & V_{1,2} & V_{2,3} & V_{3,4} & \cdots & \cdots & V_{n-1,n} & \\
 & V_{1,3} & V_{2,4} & V_{3,5} & \cdots & V_{n-2,n} & & \\
 & \cdots & & & & & & \\
 & V_{1,n-1} & V_{2,n} & & & & & \\
 & V_{1,n} & & & & & &
\end{array}$$

- For example to compute $V_{2,4}$ one needs $V_{2,2}$ and $V_{3,4}$, and then $V_{2,3}$ and $V_{4,4}$ all of which are computed earlier!

The CYK Algorithm

```
for i=1 to n do // Initialization
  $V_{i,i} = \{A \mid A \to a \text{ is a rule and } w_{i,i} = a]$
for j=2 to n do
  for i=1 to n-j+1 do
    begin
      $V_{i,j} = \{\}$; // Set $V_{i,j}$ to empty set
      for k=i to j-1 do
        $V_{i,j} = V_{i,j} \cup \{A \mid A \to BC \text{ is a rule and}$
                  $B \in V_{i,k} \text{ and } C \in V_{k+1,j}\}$
```

- This algorithm has 3 nested loops with the bound for each being $O(n)$. So the overall time/work is $O(n^3)$.
- The size of the grammar factors in as a constant factor as it is independent of n – the length of the string.
- Certain special CFGs have subcubic recognition algorithms.

The CYK Algorithm in Action

- Consider the following grammar in CNF

$$\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & BB \mid a \\ B & \rightarrow & AB \mid b \end{array}$$

- The input string is $w = aabbb$
-

$i \rightarrow$	1	2	3	4	5
	a	a	b	b	b
	$\{A\}$	$\{A\}$	$\{B\}$	$\{B\}$	$\{B\}$
	$\{\}$	$\{S, B\}$	$\{A\}$	$\{A\}$	
	$\{S, B\}$	$\{A\}$	$\{S, B\}$		
	$\{A\}$	$\{S, B\}$			
	$\{S, B\}$				

- Since $S \in V_{1,5}$, this string is in $L(G)$.

The CYK Algorithm in Action

- Consider the following grammar in CNF

$$\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & BB \mid a \\ B & \rightarrow & AB \mid b \end{array}$$

- Let us see how we compute $V_{2,4}$
  - We need to look at $V_{2,2}$ and $V_{3,4}$
  - We need to look at $V_{2,3}$ and $V_{4,4}$

$i \rightarrow$	1	2	3	4	5
	a	a	b	b	b
	$\{A\}$	$\{A\}$	$\{B\}$	$\{B\}$	$\{B\}$
	$\{\}$	$\{S,B\}$	$\{A\}$	$\{A\}$	
	$\{S,B\}$	$\{A\}$	$\{S,B\}$		
	$\{A\}$	$\{S,B\}$			
	$\{S,B\}$				

A CNF Grammar for a Fragment of English

$S \rightarrow NP\ VP$

$NP \rightarrow CN \mid CN\ PP$

$VP \rightarrow CV \mid CV\ PP$

$PP \rightarrow P\ NP$

$CN \rightarrow DT\ N$

$CV \rightarrow V \mid V\ NP$

$DT \rightarrow$ a | the

$N \rightarrow$ boy | girl | flower | telescope

$V \rightarrow$ touches | likes | sees | gives

$P \rightarrow$ with | to

Grammar in Chomsky Normal Form

$S \rightarrow NP\ VP$

$NP \rightarrow CN\ PP$

$NP \rightarrow DT\ N$

$VP \rightarrow CV\ PP$

$VP \rightarrow V\ NP$

$VP \rightarrow$ touches | likes | sees | gives

$PP \rightarrow P\ NP$

$CN \rightarrow DT\ N$

$CV \rightarrow V\ NP$

$CV \rightarrow$ touches | likes | sees | gives

$DT \rightarrow$ a | the

$N \rightarrow$ boy | girl | flower | telescope

$V \rightarrow$ touches | likes | sees | gives

$P \rightarrow$ with | to

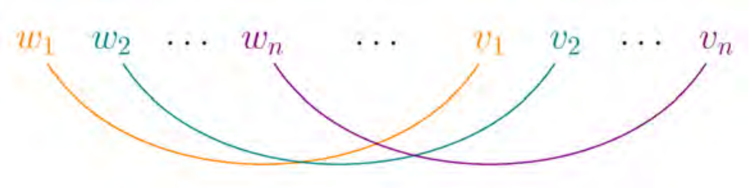
English Parsing Example with CYK

S → *NP VP*
NP → *CN PP*
NP → *DT N*
VP → *CV PP*
VP → *V NP*
VP → touches | likes | sees | gives
PP → *P NP*
CN → *DT N*
CV → *V NP*
CV → touches | likes | sees | gives
DT → a | the
N → boy | girl | flower | telescope
V → touches | likes | sees | gives
P → with | to

$i \rightarrow$	1	2	3	4	5
	the	*boy*	*sees*	*a*	*girl*
	$\{DT\}$	$\{N\}$	$\{V, CV, VP\}$	$\{DT\}$	$\{N\}$
	$\{CN, NP\}$	$\{\}$	$\{\}$	$\{CN, NP\}$	
	$\{S\}$	$\{\}$	$\{CV, VP\}$		
	$\{\}$	$\{\}$			
	$\{S\}$✓				

Some Languages are NOT Context-free

- $L = \{a^n b^n c^n \mid n \geq 0\}$ is not a context-free language.
  - This can be shown with the *Pumping Lemma for Context-free Languages*.
  - It is however a *context-sensitive language*.
- Cross-serial Dependencies[1]



- $L = \{a^n b^m c^n d^m \mid n, m \geq 0\}$ is not a context-free language but is considered *mildly context sensitive*.
  - So is $L = \{xa^n yb^m zc^n wd^m u \mid n, m \geq 0\}$

[1] Graphics by Christian Nassif-Haynes from commons.wikimedia.org/w/index.php?curid=28274222

Are CFGs enough to model natural languages?

- Swiss German has the following construct:
  dative-NP$_p$ accusative-NP$_q$ dative-taking-V$_p$ accusative-taking-V$_q$

- Jan säit das mer em Hans es huus hälfed aastriiche.
- Jan says that we Hans the house helped paint.
- “Jan says that we helped Hans paint the house.”

- Jan säit das mer d’chind em Hans es huus haend wele laa hälfe aastriiche.
- Jan says that we the children Hans the house have wanted to let help paint.
- “Jan says that we have wanted to let the children help Hans paint the house.”

Is Swiss German Context-free?

- $L_1 = \{$ Jan säit das mer (d'chind)* (em Hans)* es huus haend wele (laa)* (hälfe)* aastriiche.$\}$
- $L_2 = \{$ Swiss German $\}$
- $L_1 \cap L_2 = \{$ Jan säit das mer (d'chind)^n (em Hans)^m es huus haend wele (laa)^n (hälfe)^m aastriiche.$\} \equiv L = \{xa^n yb^m zc^n wd^m u \mid n \geq 0\}$

English “Respectively” Construct

- Alice, Bob and Carol will have a juice, a tea and a coffee, respectively.
- Again mildly context-sensitive!

Closing Remarks

- Natural languages are mildly context sensitive.
- But CFGs might be enough
- But RGs might be enough
  - If you have very big grammars and,
  - don’t really care about parsing.

11-411
Natural Language Processing
Treebanks and
Probabilistic Parsing

Kemal Oflazer

Carnegie Mellon University in Qatar

Probabilistic Parsing with CFGs

- The basic CYK Algorithm is not probabilistic: It builds a table from which all (potentially exponential number of) parse trees can be extracted.
  - Note that while computing the table needs $O(n^3)$ work, computing all trees could require exponential work!
  - Computing all trees is not necessarily useful either. How do you know which one is the correct or best tree?
- We need to incorporate probabilities in some way.
- But where do we get them?

Probabilistic Context-free Grammars

- A probabilistic CFG (PCFG) is a CFG
  - A set of nonterminal symbols $\mathcal{V}$
  - A set of terminal symbols Σ
  - A set $\mathcal{R}$ of rules of the sort $X \to Y$ where $X \in \mathcal{V}$ and $Y \in (\mathcal{V} \cup \Sigma)^*$.
  - If you need to use CKY, Chomsky Normal Form is a special case with rules only like
    - $X \to YZ$
    - $X \to a$

    where $X, Y, Z \in \mathcal{V}$ and $a \in \Sigma$

  with a probability distribution over the rules:
- For each $X \in V$, there is a probability distribution over the rules in $\mathcal{R}$, where X is the left-hand side $p(X \to Y)$
- For every X

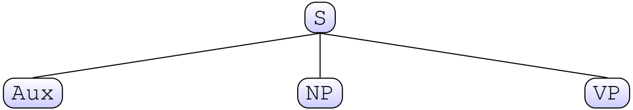
$$\sum_{X \to Y \in \mathcal{R}} p(X \to Y) = 1$$

PCFG Example

Write down the start Symbol S

Score:

PCFG Example

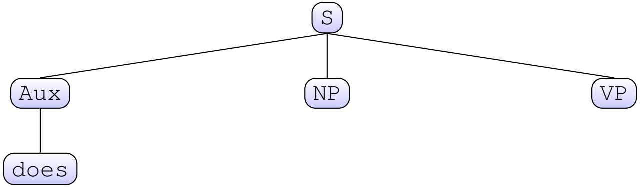


Choose a rule from the $\texttt{S}$ distribution. Here $\texttt{S} \rightarrow \texttt{Aux} \;\; \texttt{NP} \;\; \texttt{VP}$

Score:

$$p(\texttt{Aux NP VP} \mid \texttt{S})$$

PCFG Example

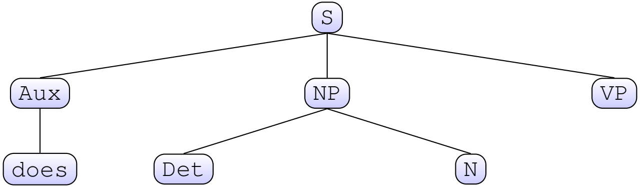


Choose a rule from the Aux distribution. Here Aux → does

Score:

$$p(\texttt{Aux NP VP} \mid \texttt{S}) \cdot p(\texttt{does} \mid \texttt{Aux})$$

PCFG Example

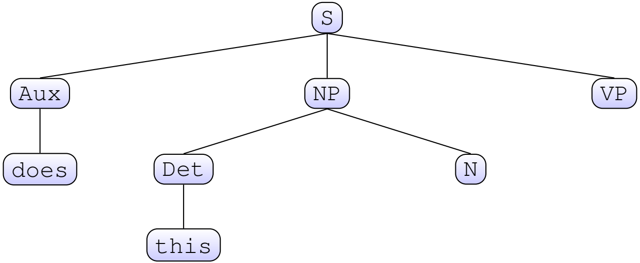


Choose a rule from the NP distribution. Here NP → Det Noun

Score:

$$p(\texttt{Aux NP VP} \mid \texttt{S}) \cdot p(\texttt{does} \mid \texttt{Aux}) \cdot p(\texttt{Det N} \mid \texttt{NP})$$

PCFG Example

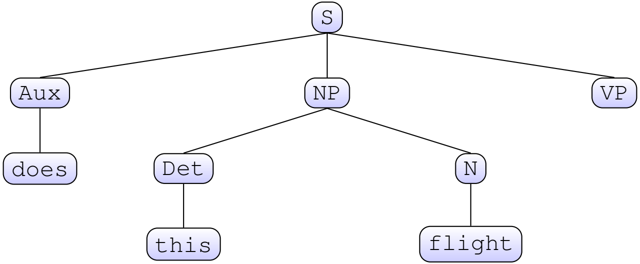


Choose a rule from the $\texttt{Det}$ distribution. Here $\texttt{Det} \rightarrow \texttt{this}$

Score:

$$p(\texttt{Aux}\,\texttt{NP}\,\texttt{VP} \mid \texttt{S}) \cdot p(\texttt{does} \mid \texttt{Aux}) \cdot p(\texttt{Det}\,\texttt{N} \mid \texttt{NP}) \cdot p(\texttt{this} \mid \texttt{Det})$$

PCFG Example



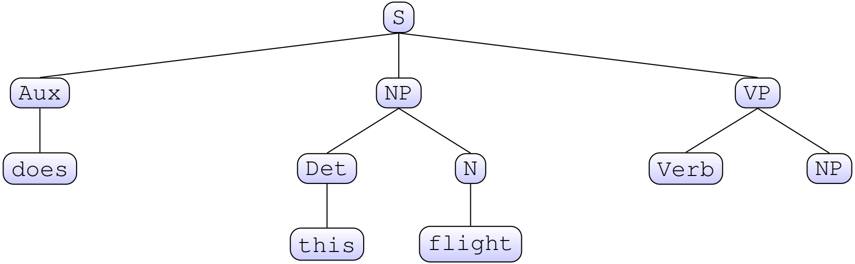
Choose a rule from the `Det` distribution. Here `Det` → `this`

Score:

$$p(\texttt{Aux NP VP} \mid \texttt{S}) \cdot p(\texttt{does} \mid \texttt{Aux}) \cdot p(\texttt{Det N} \mid \texttt{NP}) \cdot p(\texttt{this} \mid \texttt{Det}) \cdot$$

$$p(\texttt{flight} \mid \texttt{N})$$

PCFG Example



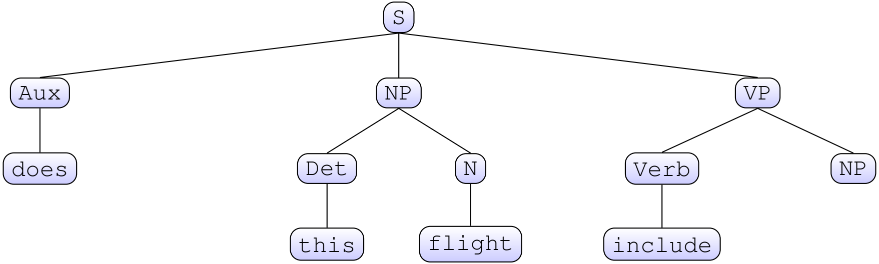
Choose a rule from the VP distribution. Here VP → Verb NP

Score:

$$p(\texttt{Aux NP VP} \mid \texttt{S}) \cdot p(\texttt{does} \mid \texttt{Aux}) \cdot p(\texttt{Det N} \mid \texttt{NP}) \cdot p(\texttt{this} \mid \texttt{Det}) \cdot$$

$$p(\texttt{flight} \mid \texttt{N}) \cdot p(\texttt{Verb NP} \mid \texttt{VP})$$

PCFG Example



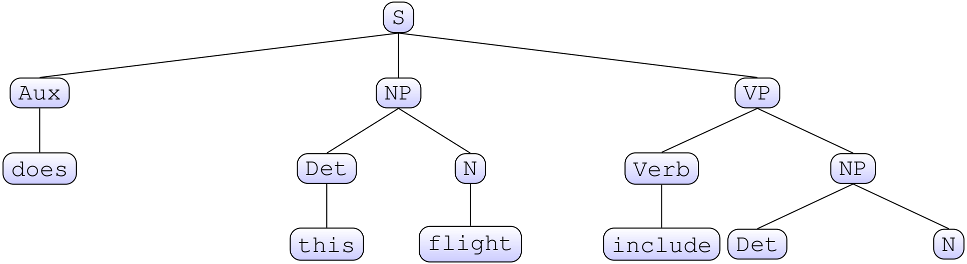
Choose a rule from the `Verb` distribution. Here `Verb` → `include`

Score:

$$p(\texttt{Aux NP VP} \mid \texttt{S}) \cdot p(\texttt{does} \mid \texttt{Aux}) \cdot p(\texttt{Det N} \mid \texttt{NP}) \cdot p(\texttt{this} \mid \texttt{Det}) \cdot$$

$$p(\texttt{flight} \mid \texttt{N}) \cdot p(\texttt{Verb NP} \mid \texttt{VP}) \cdot p(\texttt{include} \mid \texttt{V})$$

PCFG Example



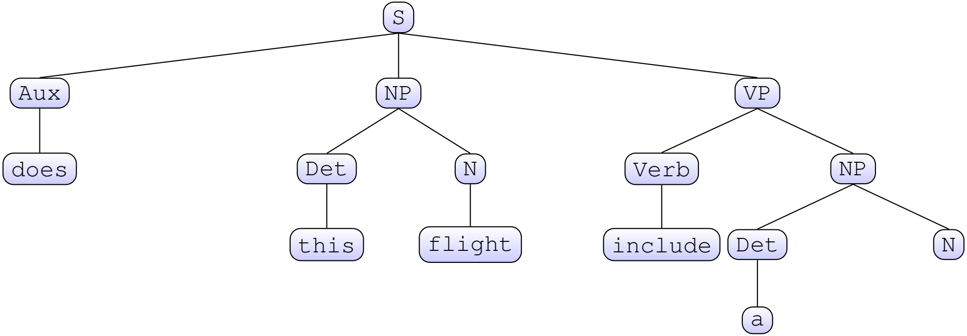
Choose a rule from the NP distribution. Here NP → Det NP

Score:

$$p(\texttt{Aux NP VP} \mid \texttt{S}) \cdot p(\texttt{does} \mid \texttt{Aux}) \cdot p(\texttt{Det N} \mid \texttt{NP}) \cdot p(\texttt{this} \mid \texttt{Det}) \cdot$$

$$p(\texttt{flight} \mid \texttt{N}) \cdot p(\texttt{Verb NP} \mid \texttt{VP}) \cdot p(\texttt{include} \mid \texttt{V}) \cdot p(\texttt{Det N} \mid \texttt{NP})$$

PCFG Example



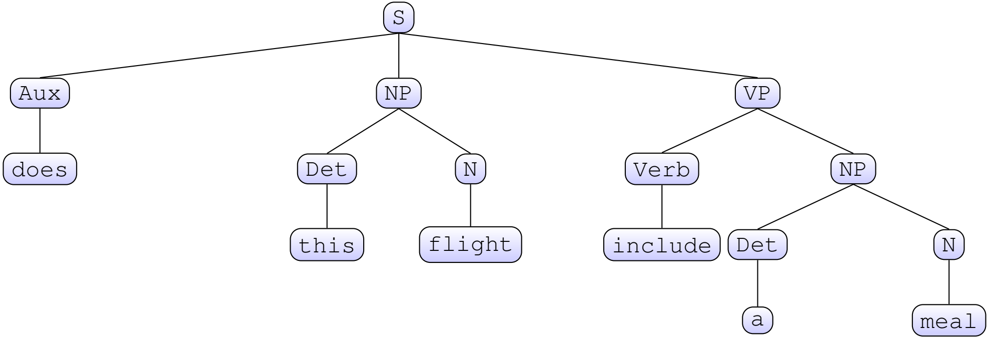
Choose a rule from the `Det` distribution. Here `Det` → `a`

Score:

$$p(\texttt{Aux NP VP} \mid \texttt{S}) \cdot p(\texttt{does} \mid \texttt{Aux}) \cdot p(\texttt{Det N} \mid \texttt{NP}) \cdot p(\texttt{this} \mid \texttt{Det}) \cdot$$

$$p(\texttt{flight} \mid \texttt{N}) \cdot p(\texttt{Verb NP} \mid \texttt{VP}) \cdot p(\texttt{include} \mid \texttt{V}) \cdot p(\texttt{Det N} \mid \texttt{NP}) \cdot p(\texttt{a} \mid \texttt{Det})$$

PCFG Example



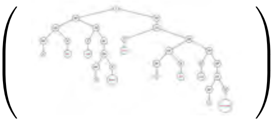
Choose a rule from the N distribution. Here N → meal

Score:

$$p(\texttt{Aux NP VP} \mid \texttt{S}) \cdot p(\texttt{does} \mid \texttt{Aux}) \cdot p(\texttt{Det N} \mid \texttt{NP}) \cdot p(\texttt{this} \mid \texttt{Det}) \cdot p(\texttt{flight} \mid \texttt{N})$$

$$p(\texttt{Verb NP} \mid \texttt{VP}) \cdot p(\texttt{include} \mid \texttt{V}) \cdot p(\texttt{Det N} \mid \texttt{NP}) \cdot p(\texttt{a} \mid \texttt{Det}) \cdot p(\texttt{meal} \mid \texttt{N})$$

Noisy Channel Model of Parsing



$\boxed{\text{Source}} \rightarrow T \rightarrow \boxed{\text{Vocal Tract/Typing}} \rightarrow X$

(A boy with a flower sees . . .)

- "I have a tree of the sentence I want to utter in my mind; by the time I utter it only the words come our."
- The PCFG defines the source model.
- The channel is deterministic: it erases everything except the leaves!
- If I observe a sequence of words comprising a sentence, what is the best tree structure it corresponds to?
- Find tree $\hat{\boldsymbol{t}} = \underset{\substack{\text{Trees } \boldsymbol{t} \\ \text{with yield } \boldsymbol{x}}}{\arg\max} \; p(\boldsymbol{t} \mid \boldsymbol{x})$
- How do we set the probabilities $p(\texttt{right hand side} \mid \texttt{left hand side})$?
- How do we decode/parse?

Probabilistic CYK

- Input
    - a PCFG $(\mathcal{V}, S, \Sigma, \mathcal{R}, p(* \mid *))$ in Chomsky Normal Form.
    - a sentence $\boldsymbol{x}$ of length n words.

- Output
    - $\hat{\boldsymbol{t}} = \arg\max_{t \in T_x} p(\boldsymbol{t} \mid \boldsymbol{x})$ (if $\boldsymbol{x}$ is in the language of the grammar.)
        - T_x: all trees with yield $\boldsymbol{x}$.

Probabilistic CYK

- We define $s_{i:j}(V)$ as the maximum probability for deriving the fragment $\ldots x_i, \ldots, x_j \ldots$ from the nonterminal $V \in \mathcal{V}$.
- We use CYK dynamic programming to compute the best score $s_{1:n}(S)$.
- **Base case**: for $i \in \{1, \ldots, n\}$ and for each $V \in \mathcal{V}$:

$$s_{i:i}(V) = p(x_i \mid V)$$

- **Inductive case**: For each $i, j, 1 \leq i < j \leq n$ and $V \in \mathcal{V}$.

$$s_{i:j}(V) = \max_{L,R \in \mathcal{V},\ i \leq k < j} \quad p(L\,R \mid V) \cdot s_{i:k}(L) \cdot s_{(k+1):j}(R)$$

- **Solution**:

$$s_{1:n}(S) = \max_{\boldsymbol{t} \in T_x} p(\boldsymbol{t})$$

Parse Chart

$i \rightarrow$	1	2	3	4	5
	the	*boy*	*sees*	*a*	*girl*
	$s_{1:1}(*)$	$s_{2:2}(*)$	$s_{3:3}(*)$	$s_{4:4}(*)$	$s_{5:5}(*)$
	$s_{1:2}(*)$	$s_{2:3}(*)$	$s_{3:4}(*)$	$s_{4:5}(*)$	
	$s_{1:3}(*)$	$s_{2:4}(*)$	$s_{3:5}(*)$		
	$s_{1:4}(*)$	$s_{2:5}(*)$			
	$s_{1:5}(*)$				

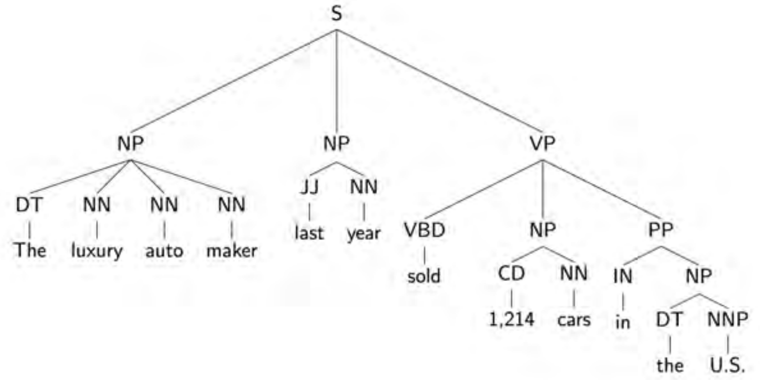
- Again, each entry is a table, mapping each nonterminal V to $s_{i:j}(V)$, the maximum probability for deriving the fragment $\ldots x_i, \ldots, x_j \ldots$ from the nonterminal V.

Remarks

- Work and Space requirements? $O(|\mathcal{R}|n^3)$ work, $O(|\mathcal{V}|n^2)$ space.
- Recovering the best tree? Use backpointers.
  - Note that there may be an exponential number of possible trees, *if you want to enumerate some/all trees*.
- Probabilistic Earley's Algorithm does NOT require a Chomsky Normal Form grammar.

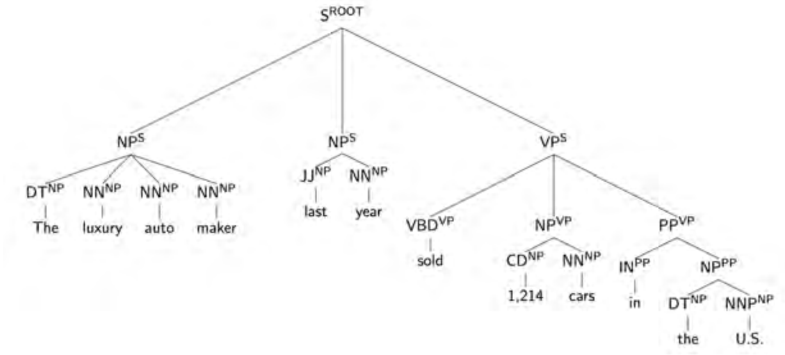
More Refined Models

Starting Point



More Refined Models

Parent Annotation

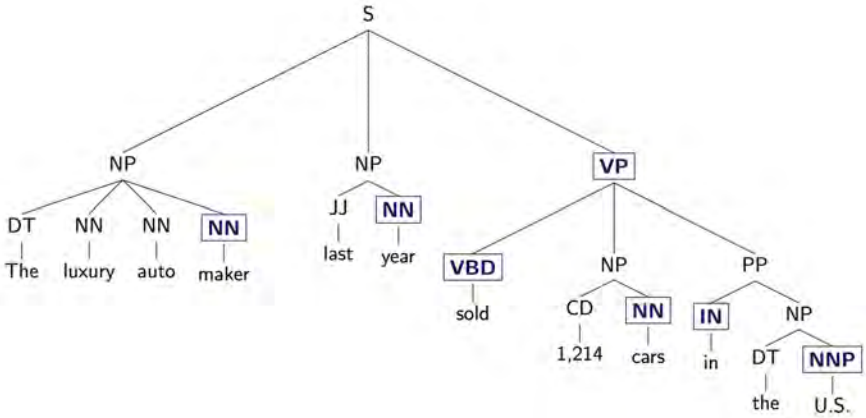


- Increase the “vertical” Markov Order

$$p(\texttt{children} \mid \texttt{parent}, \texttt{grandparent})$$

More Refined Models

Headedness

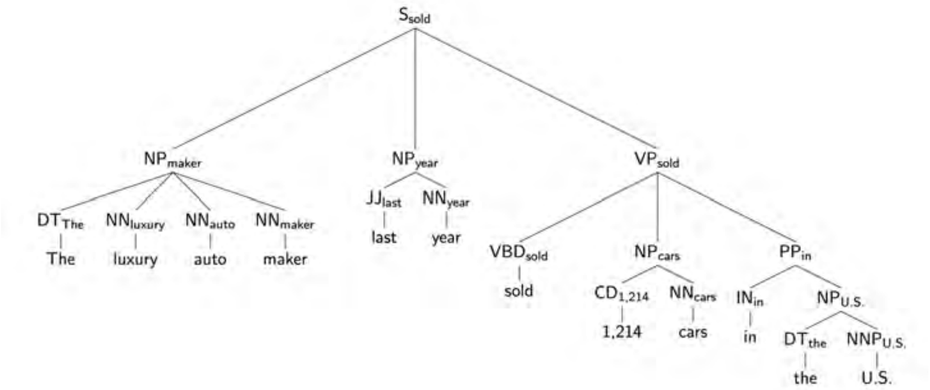


- Suggests "horizontal" Markovization:

$$p(\texttt{children} \mid \texttt{parent}) = p(\texttt{head} \mid \texttt{parent}) \cdot \prod_i p(i^{th}\ \texttt{sibling} \mid \texttt{head}, \texttt{parent})$$

More Refined Models

Lexicalization



- Each node shares a **lexical head** with its head child.

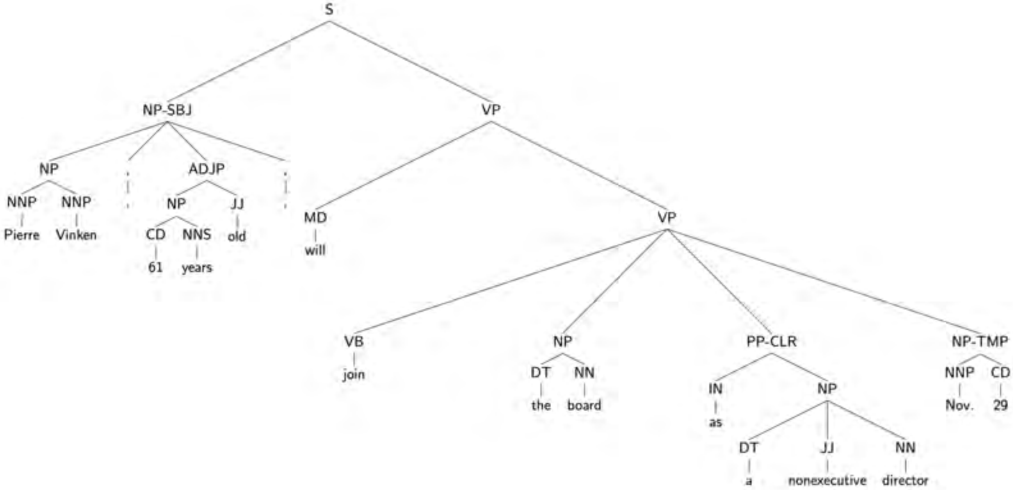
Where do the Probabilities Come from?

- Building a CFG for a natural language by hand is really hard.
  - One needs lots of categories to make sure all and only grammatical sentences are included.
  - Categories tend to start exploding combinatorially.
  - Alternative grammar formalisms are typically used for manual grammar construction; these are often based on constraints and a powerful algorithmic tool called *unification*.
- Standard approach today is to build a large-scale **treebank**, a database manually constructed parse-trees of real-world sentences.
  - Extract rules from the treebank.
  - Estimate probabilities from the treebank.

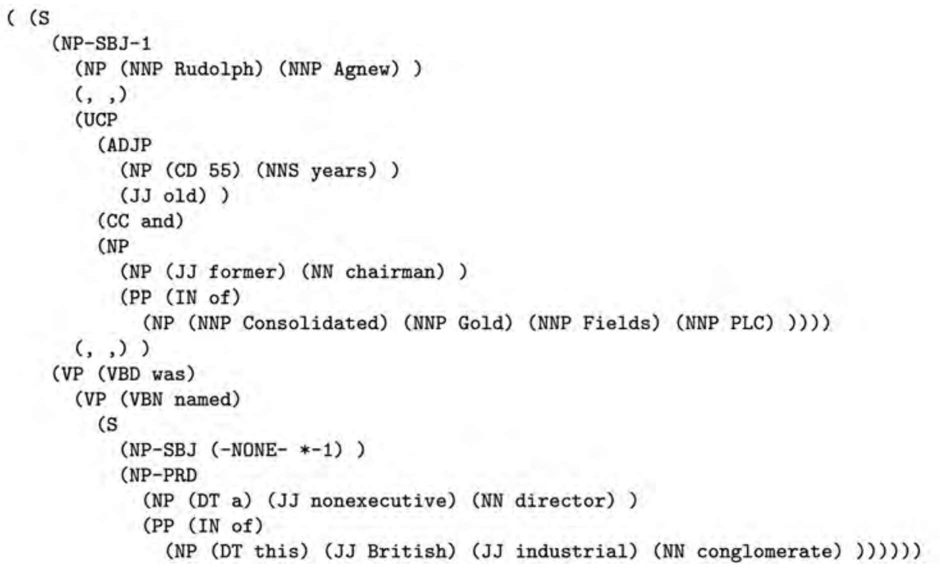
Penn Treebank

- Large database of hand-annotated parse trees of English.
- Mostly Wall Street Journal news text.
- About 42,500 sentences: typically about 40,000 used for statistical modeling and training and 2500 for testing
- WSJ section has about ≈1M words, ≈ 1M non-lexical rules boiling down to 17,500 distinct rules.
- `https://en.wikipedia.org/wiki/Treebank` lists tens of treebanks built for many different languages over the last two decades.
- `http://universaldependencies.org/` lists tens of treebanks built using the *Universal Dependencies* framework.

Example Sentence from Penn Treebank

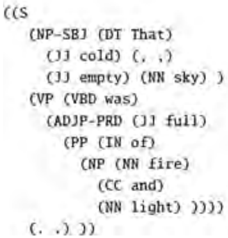


Example Sentence Encoding from Penn Treebank

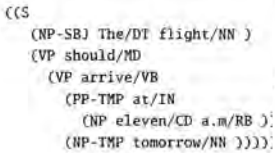


```
( (S
    (NP-SBJ-1
      (NP (NNP Rudolph) (NNP Agnew) )
      (, ,)
      (UCP
        (ADJP
          (NP (CD 55) (NNS years) )
          (JJ old) )
        (CC and)
        (NP
          (NP (JJ former) (NN chairman) )
          (PP (IN of)
            (NP (NNP Consolidated) (NNP Gold) (NNP Fields) (NNP PLC) ))))
      (, ,) )
    (VP (VBD was)
      (VP (VBN named)
        (S
          (NP-SBJ (-NONE- *-1) )
          (NP-PRD
            (NP (DT a) (JJ nonexecutive) (NN director) )
            (PP (IN of)
              (NP (DT this) (JJ British) (JJ industrial) (NN conglomerate) ))))))
```

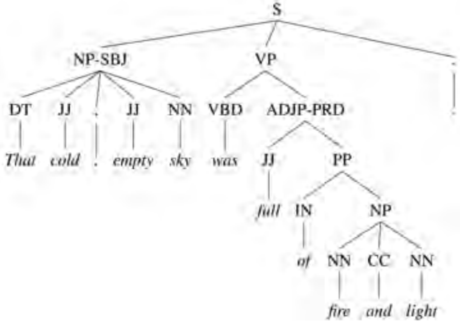
More PTB Trees



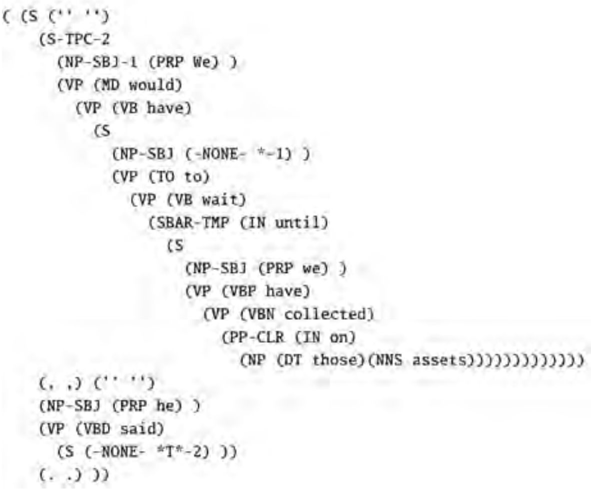
```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```



```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB )
       (NP-TMP tomorrow/NN ))))
```

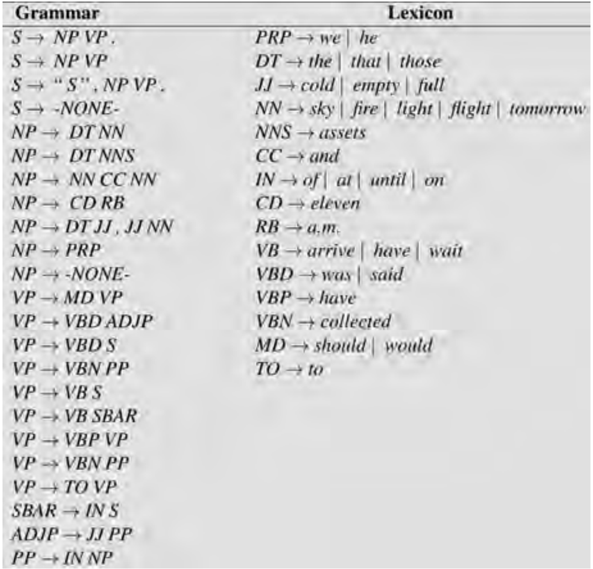


More PTB Trees



```
( (S (`` ``)
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets)))))))))))))
    (, ,) ('' '')
    (NP-SBJ (PRP he) )
    (VP (VBD said)
      (S (-NONE- *T*-2) ))
    (. .) ))
```

Treebanks as Grammars



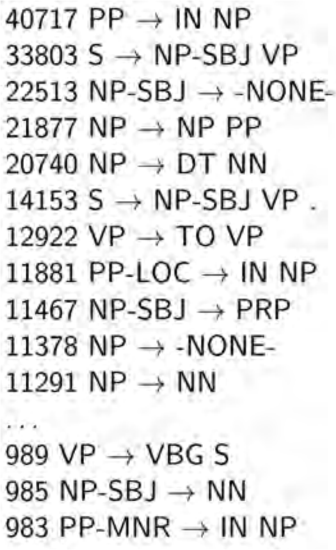
Grammar	Lexicon
S → NP VP .	PRP → we | he
S → NP VP	DT → the | that | those
S → " S " , NP VP .	JJ → cold | empty | full
S → -NONE-	NN → sky | fire | light | flight | tomorrow
NP → DT NN	NNS → assets
NP → DT NNS	CC → and
NP → NN CC NN	IN → of | at | until | on
NP → CD RB	CD → eleven
NP → DT JJ , JJ NN	RB → a.m.
NP → PRP	VB → arrive | have | wait
NP → -NONE-	VBD → was | said
VP → MD VP	VBP → have
VP → VBD ADJP	VBN → collected
VP → VBD S	MD → should | would
VP → VBN PP	TO → to
VP → VB S	
VP → VB SBAR	
VP → VBP VP	
VP → VBN PP	
VP → TO VP	
SBAR → IN S	
ADJP → JJ PP	
PP → IN NP	

- You can now compute rule probabilities from the counts of these rules e.g.,
  - $p(\texttt{VBN PP} \mid \texttt{VP})$, or
  - $p(\texttt{light} \mid \texttt{NN})$.

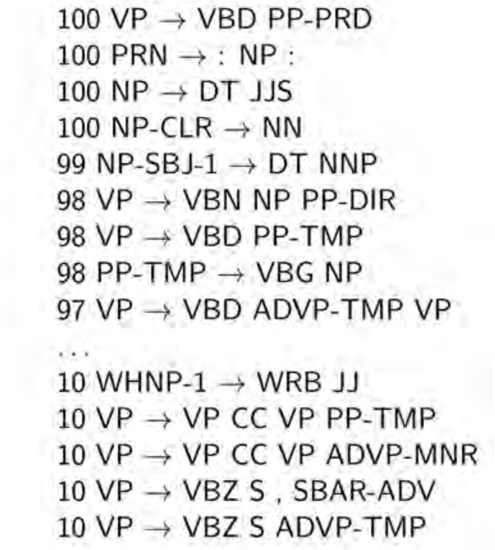
Interesting PTB Rules

- VP → VBP PP PP PP PP PP ADVP PP
  - This mostly happens because we go from football in the fall to lifting in the winter to football again in the spring.
- NP → DT JJ JJ VBG NN NNP NNP FW NNP
  - The state-owned industrial holding company Instituto Nacional de Industria . . .

Some Penn Treebank Rules with Counts



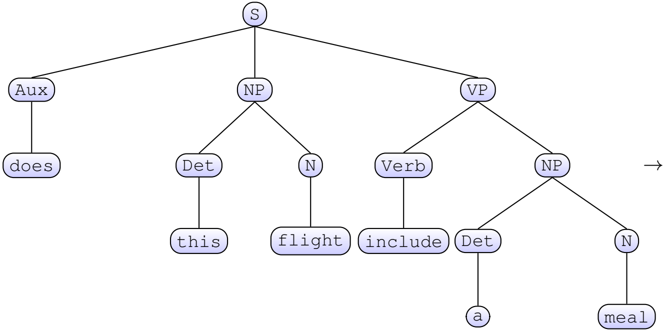
40717 PP → IN NP
33803 S → NP-SBJ VP
22513 NP-SBJ → -NONE-
21877 NP → NP PP
20740 NP → DT NN
14153 S → NP-SBJ VP .
12922 VP → TO VP
11881 PP-LOC → IN NP
11467 NP-SBJ → PRP
11378 NP → -NONE-
11291 NP → NN
. . .
989 VP → VBG S
985 NP-SBJ → NN
983 PP-MNR → IN NP



100 VP → VBD PP-PRD
100 PRN → : NP :
100 NP → DT JJS
100 NP-CLR → NN
99 NP-SBJ-1 → DT NNP
98 VP → VBN NP PP-DIR
98 VP → VBD PP-TMP
98 PP-TMP → VBG NP
97 VP → VBD ADVP-TMP VP
. . .
10 WHNP-1 → WRB JJ
10 VP → VP CC VP PP-TMP
10 VP → VP CC VP ADVP-MNR
10 VP → VBZ S , SBAR-ADV
10 VP → VBZ S ADVP-TMP

Parser Evaluation

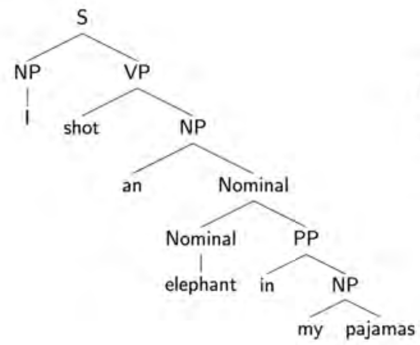
- Represent a parse tree as a collection of tuples $\{(\ell_1, i_1, j_1), (\ell_2, i_2, j_2), \ldots, (\ell_m, i_m, j_m)\}$ where
  - ℓ_k is the nonterminal labeling k^{th} phrase.
  - i_k is the index of the first word in the k^{th} phrase.
  - j_k is the index of the last word in the k^{th} phrase.

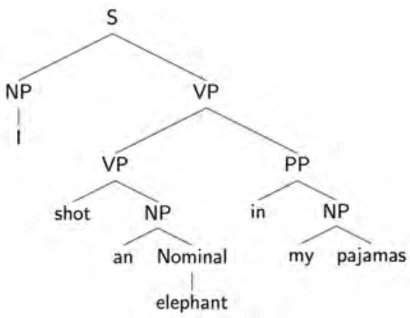


$$\rightarrow \{(S,1,6),(NP,2,3),(VP,4,6),\ldots,$$
$$(Aux,1,1),\ldots,(Noun,6,6)\}$$

- Convert gold-standard tree and system hypothesized tree into this representation, then estimate precision, recall, and F_1.

Tree Comparison Example





- In both trees: $\{(NP,1,1),(S,1,7),(VP,2,7),(PP,5,7),(NP,6,7),(Nominal,4,4)\}$
- In the left (hypothesized) tree: $\{(NP,3,7),(Nominal,4,7)\}$
- In the right (gold) tree: $\{(VP,2,4),(NP,3,4)\}$
- $P = 6/8$, $R = 6/8$

11-411
Natural Language Processing
Earley Parsing

Kemal Oflazer

Carnegie Mellon University in Qatar

Earley Parsing

- Remember that CKY parsing works only for grammar in Chomsky Normal Form (CNF)
    - Need to convert grammar to CNF.
    - The structure may not necessarily be “natural”.
    - CKY is bottom-up – may be doing unnecessary work.
- Earley algorithm allows arbitrary CFGs.
    - So no need to convert your grammar.
- Earley algorithm is a top-down algorithm.

Earley Parsing

- The Earley parser fills a table (sometimes called a *chart*) in a single sweep over the input.
- For an n word sentence, the table is of size $n + 1$.
- Table entries represent
  - In-progress constituents
  - Predicted constituents.
  - Completed constituents and their locations in the sentence

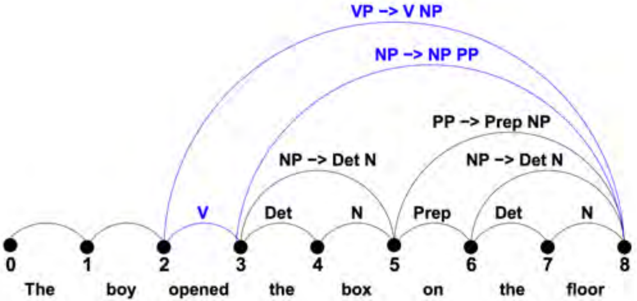


Table Entries

- Table entries are called states and are represented with dotted-rules.

$\texttt{S} \rightarrow \bullet\texttt{VP}$	a $\texttt{VP}$ is predicted
$\texttt{NP} \rightarrow \texttt{Det} \bullet \texttt{Nominal}$	an $\texttt{NP}$ is in progress
$\texttt{VP} \rightarrow \texttt{V}\ \texttt{NP}\bullet$	a VP has been found

States and Locations

$\texttt{S} \rightarrow \bullet\texttt{VP}[0,0]$ a VP is predicted at the start of the sentence

$\texttt{NP} \rightarrow \texttt{Det} \bullet \texttt{Nominal}[1,2]$ an NP is in progress; Det goes from 1 to 2

$\texttt{VP} \rightarrow \texttt{V NP} \bullet [0,3]$ a VP has been found starting at 0 and ending at 3

The Early Table Layout

Column 0	Column 1	$\ldots$	Column n
States and Locations for column 0	States and Locations for column 1	States and Locations	States and Locations for column n

- Words are positioned between columns.
- w_1 is positioned between columns 0 and 1
- w_n is positioned between columns $n-1$ and n.

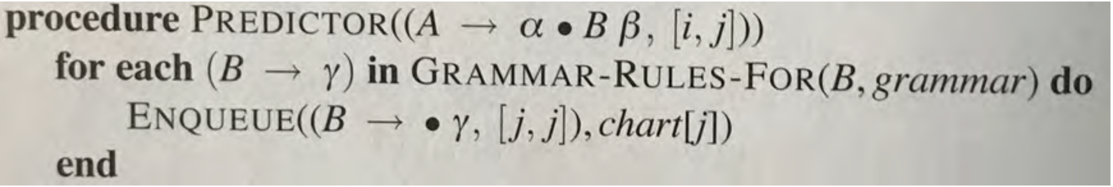
Earley – High-level Aspects

- As with most dynamic programming approaches, the answer is found by looking in the table in the right place.
- In this case, there should be an $\mathtt{S}$ state in the final column that spans from 0 to n and is complete. That is,
  - $\mathtt{S} \rightarrow \alpha \bullet \;\; [0, n]$
- If that is the case, you are done!
- So sweep through the table from 0 to n
  - New predicted states are created by starting top-down from $\mathtt{S}$
  - New incomplete states are created by advancing existing states as new constituents are discovered.
  - New complete states are created in the same way.

Earley – High-level Aspects

1. Predict all the states you can upfront
2. Read a word
   2.1 Extend states based on matches
   2.2 Generate new predictions
   2.3 Go to step 2
3. When you are out of words, look at the chart to see if you have a winner.

Earley – Main Functions: Predictor



```
procedure PREDICTOR(($A \rightarrow \alpha \bullet B\,\beta$, $[i, j]$))
    for each ($B \rightarrow \gamma$) in GRAMMAR-RULES-FOR($B$, $grammar$) do
        ENQUEUE(($B \rightarrow \bullet\gamma$, $[j, j]$), $chart[j]$)
    end
```

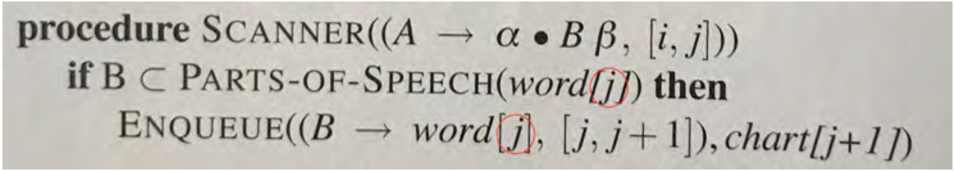
- If you have a state spanning $[i, j]$
- and is looking for a constituent B,
- then enqueue new states that will search for a B, starting at position j.

Earley– Prediction

- Given $\mathtt{A} \rightarrow \alpha \bullet \mathtt{B}\beta \quad [i,j]$ (for example $\mathtt{ROOT} \rightarrow \bullet\mathtt{S} \quad [0,0]$)
- and the rule $\mathtt{B} \rightarrow \gamma$ (for example $\mathtt{S} \rightarrow \mathtt{VP}$)
- create $\mathtt{B} \rightarrow \bullet\gamma \quad [j,j]$ (for example, $\mathtt{S} \rightarrow \bullet\mathtt{VP} \quad [0,0]$)

$\mathtt{ROOT} \rightarrow \bullet\mathtt{S}[0,0]$ $\mathtt{S} \rightarrow \bullet\mathtt{NP}\ \mathtt{VP}[0,0]$ $\mathtt{S} \rightarrow \bullet\mathtt{VP}[0,0]$ ... $\mathtt{VP} \rightarrow \bullet\mathtt{V}\ \mathtt{NP}[0,0]$ ... $\mathtt{NP} \rightarrow \bullet\mathtt{DT}\ \mathtt{N}[0,0]$			

Earley – Main Functions: Scanner



```
procedure SCANNER(($A \rightarrow \alpha \bullet B\,\beta$, $[i, j]$))
    if B $\subset$ PARTS-OF-SPEECH(word[$j$]) then
        ENQUEUE(($B \rightarrow$ word[$j$], $[j, j+1]$), chart[$j$+1])
```

Red circled indices should be $j + 1$

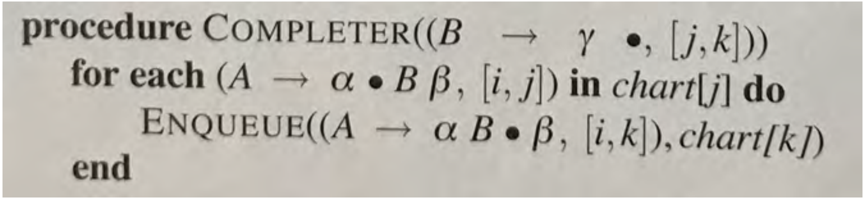
- If you have a state spanning $[i, j]$
- and is looking for a word with Part-of-Speech $\mathtt{B}$,
- and one of the parts-of-speech the next word at position j is $\mathtt{B}$
- then enqueue a state of the sort $\mathtt{B} \rightarrow \mathit{word}[j + 1] \bullet\ [j, j + 1]$ in chart position $j + 1$

Earley– Scanning

- Given $\mathtt{A} \rightarrow \alpha \bullet \mathtt{B}\beta \quad [i,j]$ (for example $\mathtt{VP} \rightarrow \bullet\mathtt{V}\ \mathtt{NP} \quad [0,0]$)
- and the rule $\mathtt{B} \rightarrow w_{j+1}$ (for example $\mathtt{V} \rightarrow \mathit{book}$)
- create $\mathtt{B} \rightarrow w_{j+1} \bullet \quad [j,j+1]$ (for example, $\mathtt{V} \rightarrow \mathit{book} \bullet \quad [0,1]$)

$\mathtt{ROOT} \rightarrow \bullet\mathtt{S}[0,0]$ $\mathtt{S} \rightarrow \bullet\mathtt{NP}\ \mathtt{VP}[0,0]$ $\mathtt{S} \rightarrow \bullet\mathtt{VP}[0,0]$ $\ldots$ $\mathtt{VP} \rightarrow \bullet\mathtt{V}\ \mathtt{NP}[0,0]$ $\ldots$ $\mathtt{NP} \rightarrow \bullet\mathtt{DT}\ \mathtt{N}[0,0]$	$\mathtt{V} \rightarrow \mathit{book} \bullet \quad [0,1]$		

Earley – Main Functions: Completer



```
procedure COMPLETER(($B \rightarrow \gamma \bullet$, $[j,k]$))
    for each ($A \rightarrow \alpha \bullet B\, \beta$, $[i,j]$) in $chart[j]$ do
        ENQUEUE(($A \rightarrow \alpha\, B \bullet \beta$, $[i,k]$), $chart[k]$)
end
```

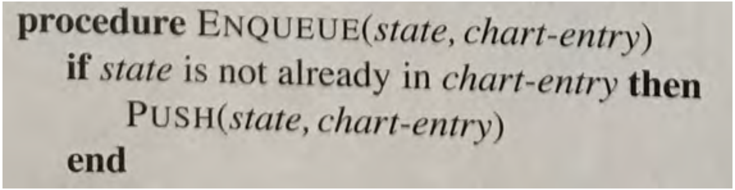
- If you have a completed state spanning $[j,k]$ with B as the left hand side.
- then, for each state in chart position j (with some span $[i,j]$, that is immediately looking for a B),
- move the dot to after B,
- extend the span to $[i,k]$
- then enqueue the updated state in chart position k.

Earley– Completion

- Given $\mathtt{A} \rightarrow \alpha \bullet \mathtt{B}\beta \quad [i, j]$ (for example $\mathtt{VP} \rightarrow \bullet \mathtt{V}\ \mathtt{NP} \quad [0, 0]$)
- and $\mathtt{B} \rightarrow \gamma \bullet \quad [j, k]$ (for example $\mathtt{V} \rightarrow \mathit{book} \bullet \quad [0, 1]$)
- create $\mathtt{V} \rightarrow \alpha\mathtt{B} \bullet \beta \quad [i, k]$ (for example, $\mathtt{VP} \rightarrow \mathtt{V} \bullet \mathtt{NP} \quad [0, 1]$)

$\mathtt{ROOT} \rightarrow \bullet \mathtt{S}[0, 0]$ $\mathtt{S} \rightarrow \bullet \mathtt{NP}\ \mathtt{VP}[0, 0]$ $\mathtt{S} \rightarrow \bullet \mathtt{VP}[0, 0]$ $\ldots$ $\mathtt{VP} \rightarrow \bullet \mathtt{V}\ \mathtt{NP}[0, 0]$ $\ldots$ $\mathtt{NP} \rightarrow \bullet \mathtt{DT}\ \mathtt{N}[0, 0]$	$\mathtt{V} \rightarrow \mathit{book} \bullet \quad [0, 1]$ $\mathtt{VP} \rightarrow \mathtt{V} \bullet \mathtt{NP} \quad [0, 1]$		

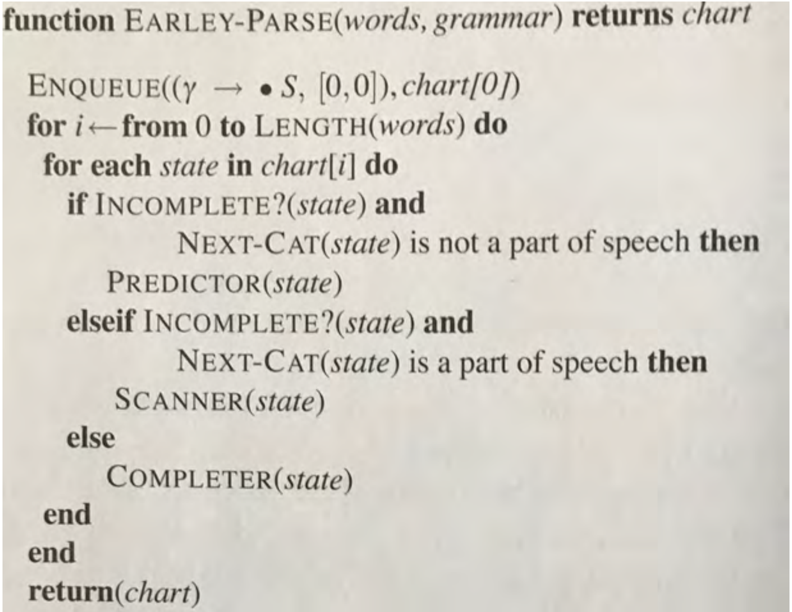
Earley – Main Functions: Enqueue



```
procedure ENQUEUE(state, chart-entry)
    if state is not already in chart-entry then
        PUSH(state, chart-entry)
    end
```

- Just enter the given state to the chart-entry if it is not already there.

The Earley Parser

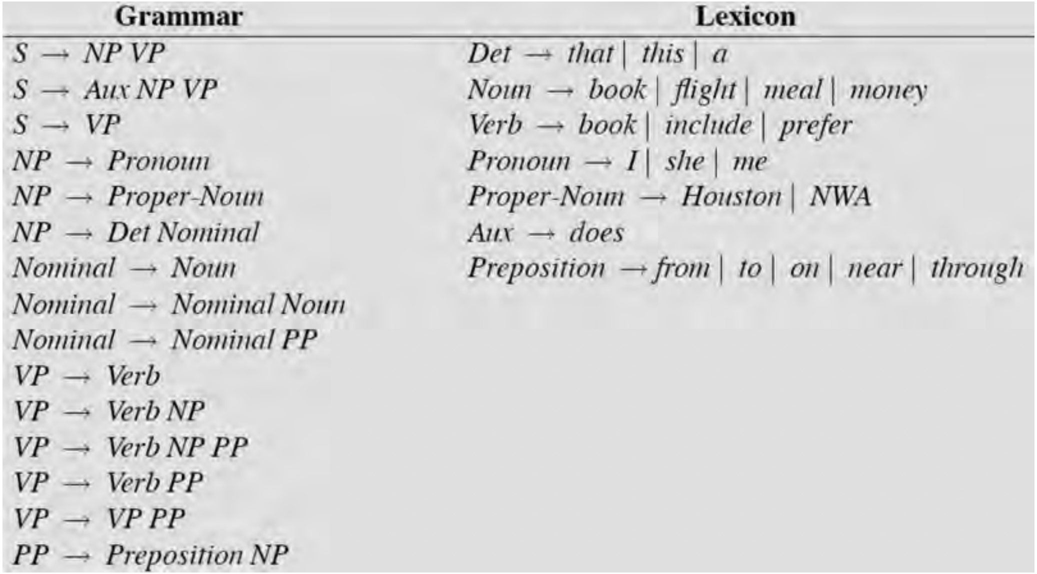


```
function EARLEY-PARSE(words, grammar) returns chart

  ENQUEUE(($\gamma \rightarrow \bullet S$, [0,0]), chart[0])
  for i ← from 0 to LENGTH(words) do
    for each state in chart[i] do
      if INCOMPLETE?(state) and
                NEXT-CAT(state) is not a part of speech then
          PREDICTOR(state)
      elseif INCOMPLETE?(state) and
                NEXT-CAT(state) is a part of speech then
          SCANNER(state)
      else
          COMPLETER(state)
    end
  end
  return(chart)
```

Extended Earley Example

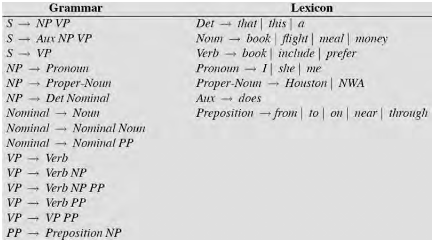
- $_0$ Book $_1$ that $_2$ flight $_3$
- We should find a completed state at chart position 3
- with left hand side S and is spanning $[0, 3]$

Extended Earley Example Grammar

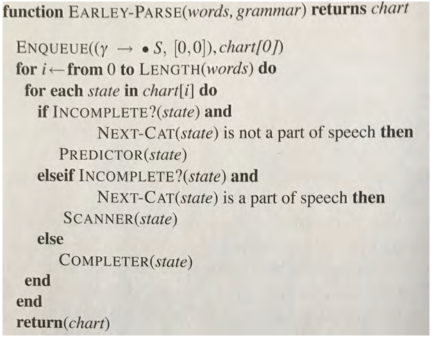


Grammar	Lexicon
S → *NP VP*	*Det* → *that* | *this* | *a*
S → *Aux NP VP*	*Noun* → *book* | *flight* | *meal* | *money*
S → *VP*	*Verb* → *book* | *include* | *prefer*
NP → *Pronoun*	*Pronoun* → *I* | *she* | *me*
NP → *Proper-Noun*	*Proper-Noun* → *Houston* | *NWA*
NP → *Det Nominal*	*Aux* → *does*
Nominal → *Noun*	*Preposition* → *from* | *to* | *on* | *near* | *through*
Nominal → *Nominal Noun*	
Nominal → *Nominal PP*	
VP → *Verb*	
VP → *Verb NP*	
VP → *Verb NP PP*	
VP → *Verb PP*	
VP → *VP PP*	
PP → *Preposition NP*	

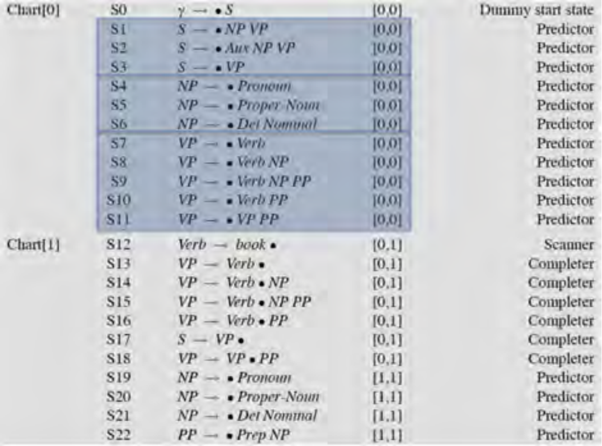
Extended Earley Example



Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

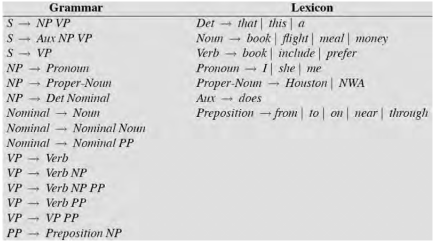


```
function EARLEY-PARSE(words, grammar) returns chart

  ENQUEUE(($\gamma \rightarrow \bullet S$, [0,0]), chart[0])
  for i ← from 0 to LENGTH(words) do
    for each state in chart[i] do
      if INCOMPLETE?(state) and
              NEXT-CAT(state) is not a part of speech then
          PREDICTOR(state)
      elseif INCOMPLETE?(state) and
              NEXT-CAT(state) is a part of speech then
          SCANNER(state)
      else
          COMPLETER(state)
    end
  end
  return(chart)
```

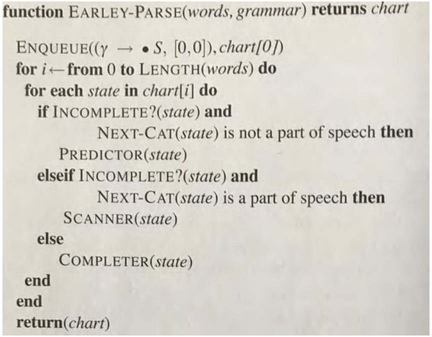


Chart[0]	S0	$\gamma \rightarrow \bullet S$	[0,0]	Dummy start state
	S1	$S \rightarrow \bullet NP\ VP$	[0,0]	Predictor
	S2	$S \rightarrow \bullet Aux\ NP\ VP$	[0,0]	Predictor
	S3	$S \rightarrow \bullet VP$	[0,0]	Predictor
	S4	$NP \rightarrow \bullet Pronoun$	[0,0]	Predictor
	S5	$NP \rightarrow \bullet Proper\text{-}Noun$	[0,0]	Predictor
	S6	$NP \rightarrow \bullet Det\ Nominal$	[0,0]	Predictor
	S7	$VP \rightarrow \bullet Verb$	[0,0]	Predictor
	S8	$VP \rightarrow \bullet Verb\ NP$	[0,0]	Predictor
	S9	$VP \rightarrow \bullet Verb\ NP\ PP$	[0,0]	Predictor
	S10	$VP \rightarrow \bullet Verb\ PP$	[0,0]	Predictor
	S11	$VP \rightarrow \bullet VP\ PP$	[0,0]	Predictor
Chart[1]	S12	$Verb \rightarrow book \bullet$	[0,1]	Scanner
	S13	$VP \rightarrow Verb \bullet$	[0,1]	Completer
	S14	$VP \rightarrow Verb \bullet NP$	[0,1]	Completer
	S15	$VP \rightarrow Verb \bullet NP\ PP$	[0,1]	Completer
	S16	$VP \rightarrow Verb \bullet PP$	[0,1]	Completer
	S17	$S \rightarrow VP \bullet$	[0,1]	Completer
	S18	$VP \rightarrow VP \bullet PP$	[0,1]	Completer
	S19	$NP \rightarrow \bullet Pronoun$	[1,1]	Predictor
	S20	$NP \rightarrow \bullet Proper\text{-}Noun$	[1,1]	Predictor
	S21	$NP \rightarrow \bullet Det\ Nominal$	[1,1]	Predictor
	S22	$PP \rightarrow \bullet Prep\ NP$	[1,1]	Predictor

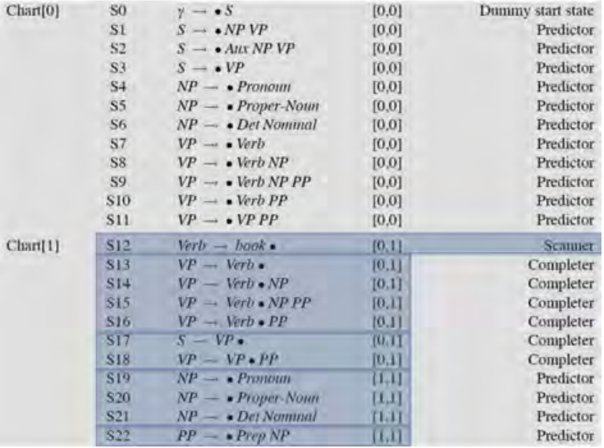
Extended Earley Example



Grammar	Lexicon
S → *NP VP*	*Det* → *that* | *this* | *a*
S → *Aux NP VP*	*Noun* → *book* | *flight* | *meal* | *money*
S → *VP*	*Verb* → *book* | *include* | *prefer*
NP → *Pronoun*	*Pronoun* → *I* | *she* | *me*
NP → *Proper-Noun*	*Proper-Noun* → *Houston* | *NWA*
NP → *Det Nominal*	*Aux* → *does*
Nominal → *Noun*	*Preposition* → *from* | *to* | *on* | *near* | *through*
Nominal → *Nominal Noun*	
Nominal → *Nominal PP*	
VP → *Verb*	
VP → *Verb NP*	
VP → *Verb NP PP*	
VP → *Verb PP*	
VP → *VP PP*	
PP → *Preposition NP*	

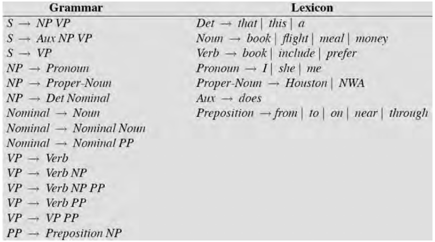


```
function EARLEY-PARSE(words, grammar) returns chart

  ENQUEUE(($\gamma \rightarrow \bullet S$, [0,0]), chart[0])
  for i ← from 0 to LENGTH(words) do
    for each state in chart[i] do
      if INCOMPLETE?(state) and
              NEXT-CAT(state) is not a part of speech then
          PREDICTOR(state)
      elseif INCOMPLETE?(state) and
              NEXT-CAT(state) is a part of speech then
          SCANNER(state)
      else
          COMPLETER(state)
    end
  end
  return(chart)
```

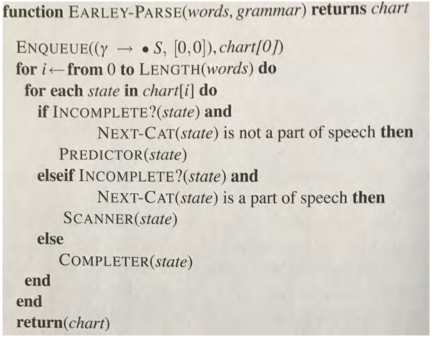


Chart[0]	S0	$\gamma \rightarrow \bullet S$	[0,0]	Dummy start state
	S1	$S \rightarrow \bullet NP\ VP$	[0,0]	Predictor
	S2	$S \rightarrow \bullet Aux\ NP\ VP$	[0,0]	Predictor
	S3	$S \rightarrow \bullet VP$	[0,0]	Predictor
	S4	$NP \rightarrow \bullet Pronoun$	[0,0]	Predictor
	S5	$NP \rightarrow \bullet Proper\text{-}Noun$	[0,0]	Predictor
	S6	$NP \rightarrow \bullet Det\ Nominal$	[0,0]	Predictor
	S7	$VP \rightarrow \bullet Verb$	[0,0]	Predictor
	S8	$VP \rightarrow \bullet Verb\ NP$	[0,0]	Predictor
	S9	$VP \rightarrow \bullet Verb\ NP\ PP$	[0,0]	Predictor
	S10	$VP \rightarrow \bullet Verb\ PP$	[0,0]	Predictor
	S11	$VP \rightarrow \bullet VP\ PP$	[0,0]	Predictor
Chart[1]	S12	$Verb \rightarrow book \bullet$	[0,1]	Scanner
	S13	$VP \rightarrow Verb \bullet$	[0,1]	Completer
	S14	$VP \rightarrow Verb \bullet NP$	[0,1]	Completer
	S15	$VP \rightarrow Verb \bullet NP\ PP$	[0,1]	Completer
	S16	$VP \rightarrow Verb \bullet PP$	[0,1]	Completer
	S17	$S \rightarrow VP \bullet$	[0,1]	Completer
	S18	$VP \rightarrow VP \bullet PP$	[0,1]	Completer
	S19	$NP \rightarrow \bullet Pronoun$	[1,1]	Predictor
	S20	$NP \rightarrow \bullet Proper\text{-}Noun$	[1,1]	Predictor
	S21	$NP \rightarrow \bullet Det\ Nominal$	[1,1]	Predictor
	S22	$PP \rightarrow \bullet Prep\ NP$	[1,1]	Predictor

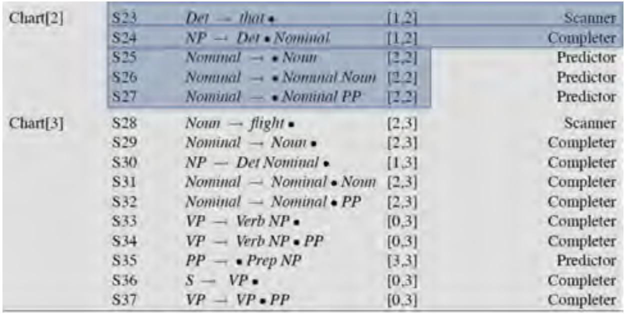
Extended Earley Example



Grammar	Lexicon
S → *NP VP*	*Det* → *that* | *this* | *a*
S → *Aux NP VP*	*Noun* → *book* | *flight* | *meal* | *money*
S → *VP*	*Verb* → *book* | *include* | *prefer*
NP → *Pronoun*	*Pronoun* → *I* | *she* | *me*
NP → *Proper-Noun*	*Proper-Noun* → *Houston* | *NWA*
NP → *Det Nominal*	*Aux* → *does*
Nominal → *Noun*	*Preposition* → *from* | *to* | *on* | *near* | *through*
Nominal → *Nominal Noun*	
Nominal → *Nominal PP*	
VP → *Verb*	
VP → *Verb NP*	
VP → *Verb NP PP*	
VP → *Verb PP*	
VP → *VP PP*	
PP → *Preposition NP*	

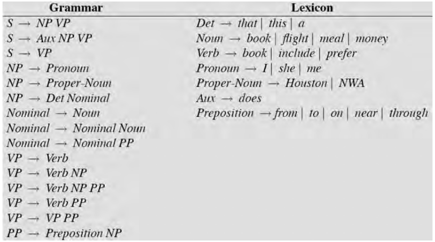


```
function EARLEY-PARSE(words, grammar) returns chart

 ENQUEUE((γ → • S, [0,0]), chart[0])
 for i ← from 0 to LENGTH(words) do
   for each state in chart[i] do
     if INCOMPLETE?(state) and
              NEXT-CAT(state) is not a part of speech then
         PREDICTOR(state)
     elseif INCOMPLETE?(state) and
              NEXT-CAT(state) is a part of speech then
          SCANNER(state)
     else
         COMPLETER(state)
   end
 end
 return(chart)
```

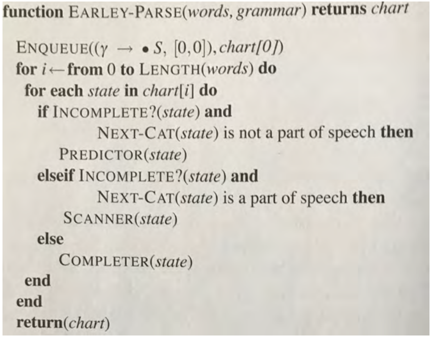


Chart[2]	S23	*Det* → *that* •	[1,2]	Scanner
	S24	*NP* → *Det* • *Nominal*	[1,2]	Completer
	S25	*Nominal* → • *Noun*	[2,2]	Predictor
	S26	*Nominal* → • *Nominal Noun*	[2,2]	Predictor
	S27	*Nominal* → • *Nominal PP*	[2,2]	Predictor
Chart[3]	S28	*Noun* → *flight* •	[2,3]	Scanner
	S29	*Nominal* → *Noun* •	[2,3]	Completer
	S30	*NP* → *Det Nominal* •	[1,3]	Completer
	S31	*Nominal* → *Nominal* • *Noun*	[2,3]	Completer
	S32	*Nominal* → *Nominal* • *PP*	[2,3]	Completer
	S33	*VP* → *Verb NP* •	[0,3]	Completer
	S34	*VP* → *Verb NP* • *PP*	[0,3]	Completer
	S35	*PP* → • *Prep NP*	[3,3]	Predictor
	S36	*S* → *VP* •	[0,3]	Completer
	S37	*VP* → *VP* • *PP*	[0,3]	Completer

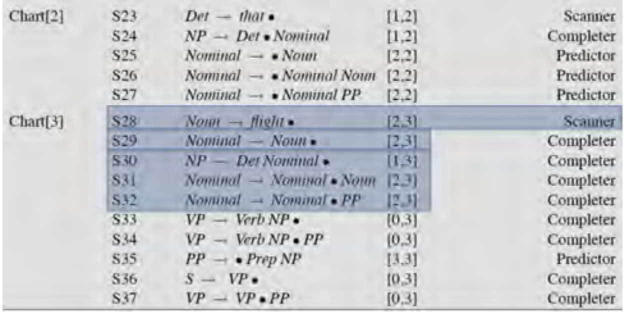
Extended Earley Example



Grammar	Lexicon
S → *NP VP*	*Det* → *that* | *this* | *a*
S → *Aux NP VP*	*Noun* → *book* | *flight* | *meal* | *money*
S → *VP*	*Verb* → *book* | *include* | *prefer*
NP → *Pronoun*	*Pronoun* → *I* | *she* | *me*
NP → *Proper-Noun*	*Proper-Noun* → *Houston* | *NWA*
NP → *Det Nominal*	*Aux* → *does*
Nominal → *Noun*	*Preposition* → *from* | *to* | *on* | *near* | *through*
Nominal → *Nominal Noun*	
Nominal → *Nominal PP*	
VP → *Verb*	
VP → *Verb NP*	
VP → *Verb NP PP*	
VP → *Verb PP*	
VP → *VP PP*	
PP → *Preposition NP*	

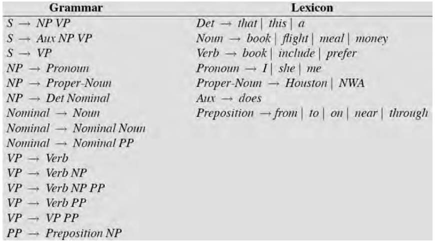


```
function EARLEY-PARSE(words, grammar) returns chart

  ENQUEUE(($\gamma \rightarrow \bullet S$, [0,0]), chart[0])
  for i ← from 0 to LENGTH(words) do
    for each state in chart[i] do
      if INCOMPLETE?(state) and
            NEXT-CAT(state) is not a part of speech then
        PREDICTOR(state)
      elseif INCOMPLETE?(state) and
            NEXT-CAT(state) is a part of speech then
        SCANNER(state)
      else
        COMPLETER(state)
    end
  end
  return(chart)
```

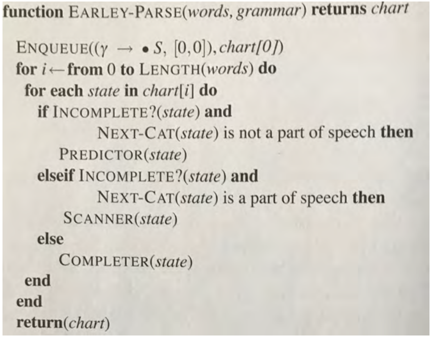


Chart[2]	S23	*Det* → *that* •	[1,2]	Scanner
	S24	*NP* → *Det* • *Nominal*	[1,2]	Completer
	S25	*Nominal* → • *Noun*	[2,2]	Predictor
	S26	*Nominal* → • *Nominal Noun*	[2,2]	Predictor
	S27	*Nominal* → • *Nominal PP*	[2,2]	Predictor
Chart[3]	S28	*Noun* → *flight* •	[2,3]	Scanner
	S29	*Nominal* → *Noun* •	[2,3]	Completer
	S30	*NP* → *Det Nominal* •	[1,3]	Completer
	S31	*Nominal* → *Nominal* • *Noun*	[2,3]	Completer
	S32	*Nominal* → *Nominal* • *PP*	[2,3]	Completer
	S33	*VP* → *Verb NP* •	[0,3]	Completer
	S34	*VP* → *Verb NP* • *PP*	[0,3]	Completer
	S35	*PP* → • *Prep NP*	[3,3]	Predictor
	S36	*S* → *VP* •	[0,3]	Completer
	S37	*VP* → *VP* • *PP*	[0,3]	Completer

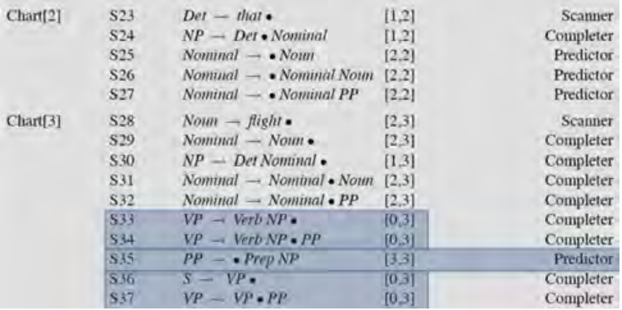
Extended Earley Example



Grammar	Lexicon
S → *NP VP*	*Det* → *that* | *this* | *a*
S → *Aux NP VP*	*Noun* → *book* | *flight* | *meal* | *money*
S → *VP*	*Verb* → *book* | *include* | *prefer*
NP → *Pronoun*	*Pronoun* → *I* | *she* | *me*
NP → *Proper-Noun*	*Proper-Noun* → *Houston* | *NWA*
NP → *Det Nominal*	*Aux* → *does*
Nominal → *Noun*	*Preposition* → *from* | *to* | *on* | *near* | *through*
Nominal → *Nominal Noun*	
Nominal → *Nominal PP*	
VP → *Verb*	
VP → *Verb NP*	
VP → *Verb NP PP*	
VP → *Verb PP*	
VP → *VP PP*	
PP → *Preposition NP*	



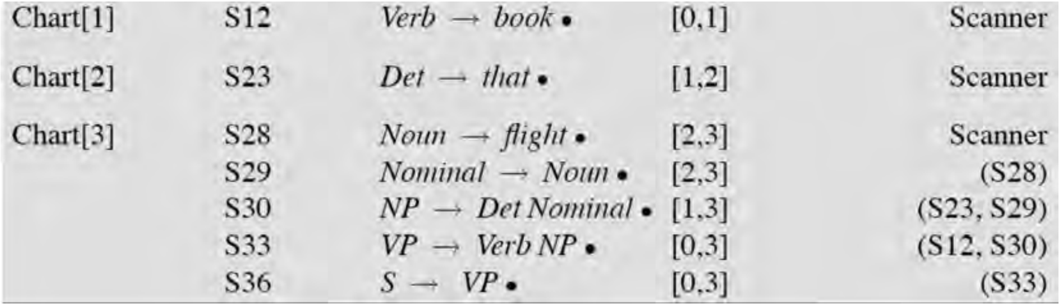
```
function EARLEY-PARSE(words, grammar) returns chart

  ENQUEUE(($\gamma \rightarrow \bullet S$, [0,0]), chart[0])
  for i ← from 0 to LENGTH(words) do
    for each state in chart[i] do
      if INCOMPLETE?(state) and
                NEXT-CAT(state) is not a part of speech then
          PREDICTOR(state)
      elseif INCOMPLETE?(state) and
                NEXT-CAT(state) is a part of speech then
          SCANNER(state)
      else
          COMPLETER(state)
    end
  end
  return(chart)
```



Chart[2]	S23	*Det* → *that* •	[1,2]	Scanner
	S24	*NP* → *Det* • *Nominal*	[1,2]	Completer
	S25	*Nominal* → • *Noun*	[2,2]	Predictor
	S26	*Nominal* → • *Nominal Noun*	[2,2]	Predictor
	S27	*Nominal* → • *Nominal PP*	[2,2]	Predictor
Chart[3]	S28	*Noun* → *flight* •	[2,3]	Scanner
	S29	*Nominal* → *Noun* •	[2,3]	Completer
	S30	*NP* → *Det Nominal* •	[1,3]	Completer
	S31	*Nominal* → *Nominal* • *Noun*	[2,3]	Completer
	S32	*Nominal* → *Nominal* • *PP*	[2,3]	Completer
	S33	*VP* → *Verb NP* •	[0,3]	Completer
	S34	*VP* → *Verb NP* • *PP*	[0,3]	Completer
	S35	*PP* → • *Prep NP*	[3,3]	Predictor
	S36	*S* → *VP* •	[0,3]	Completer
	S37	*VP* → *VP* • *PP*	[0,3]	Completer

S37 is due to S11 and S33!

Final Earley Parse



Chart[1]	S12	*Verb* → *book* •	[0,1]	Scanner
Chart[2]	S23	*Det* → *that* •	[1,2]	Scanner
Chart[3]	S28	*Noun* → *flight* •	[2,3]	Scanner
	S29	*Nominal* → *Noun* •	[2,3]	(S28)
	S30	*NP* → *Det Nominal* •	[1,3]	(S23, S29)
	S33	*VP* → *Verb NP* •	[0,3]	(S12, S30)
	S36	*S* → *VP* •	[0,3]	(S33)

Comments

- Work is $O(n^3)$ – analysis is a bit trickier than CKY.
- Space is $O(n^2)$
- Big grammar-related constant
- Backpointers can help recover trees.

Probabilistic Earley Parser

- So far we had no mention of any probabilities or choosing the best parse.
- Rule probabilities can be estimated from treebanks as usual.
- There are algorithms that resemble the Viterbi algorithm for HMMs for computing the best parse incrementally from left to right it the chart.
  - Stolcke, Andreas, “An efficient probabilistic context-free parsing algorithm that computes prefix probabilities”, Computational Linguistics, Volume 21, No:2, 1995
- Beyond our scope!

General Chart Parsing

- CKY and Earley each statically determine order of events, in code
  - CKY fills out triangle table starting with words on the diagonal
  - Earley marches through instances of grammar rules
- Chart parsing puts *edges* on an *agenda*, allowing an arbitrary ordering policy, separate from the code.
- Generalizes CKY, Earley, and others

Implementing Parsing as Search

```
Agenda = {$state_0$}
while (Agenda not empty)
    s = pop a state from Agenda
    if s a success-state return s // we have a parse
    else if s is not a failure-state:
        generate new states from s
        push new states on Agenda
return nil // no parser
```

- Fundamental Rule of Chart Parsing: if you can combine two contiguous edges to make a bigger one, do it.
- Akin to the Completer function in Earley.
- How you interact with the agenda is called a **strategy**.

Is Ambiguity Solved?

- Time flies like an arrow.
- Fruit flies like a banana.

- Time/N flies/V like an arrow.
- Time/V flies/N like (you time) an arrow.
- Time/V flies/N like an arrow (times flies).
- Time/V flies/N (that are) like an arrow.
- [Time/N flies/N] like/V an arrow!
- . . .

11-411
Natural Language Processing
Dependency Parsing

Kemal Oflazer

Carnegie Mellon University in Qatar

Dependencies

Informally, you can think of dependency structures as a transformation of phrase-structures that

- maintains the word-to-word relationships induced by lexicalization,
- adds labels to these relationships, and
- eliminates the phrase categories

There are linguistic theories build on dependencies, as well as treebanks based on dependencies:

- Czech Treebank
- Turkish Treebank

Dependency Tree: Definition

Let $\boldsymbol{x} = [x_1, \ldots, x_n]$ be a sentence. We add a special ROOT symbol as "x_0".

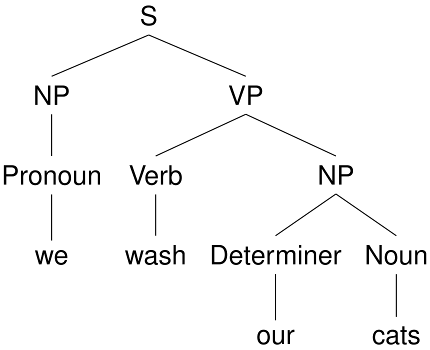
A dependency tree consists of a set of tuples $[p, c, \ell]$ where

- $p \in \{0, \ldots, n\}$ is the index of a *parent*.
- $c \in \{1, \ldots, n\}$ is the index of a *child*.
- $\ell \in \mathcal{L}$ is a label.

Different annotation schemes define different label sets $\mathcal{L}$, and different constraints on the set of tuples. Most commonly:

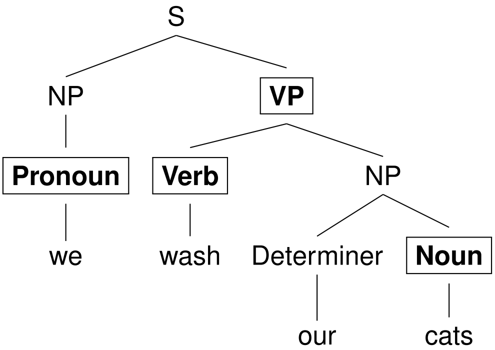
- The tuple is represented as a directed edge from x_p to x_c with label ℓ.
- The directed edges form an directed tree with x_0 as the root (sometimes denoted as ROOT).

Example



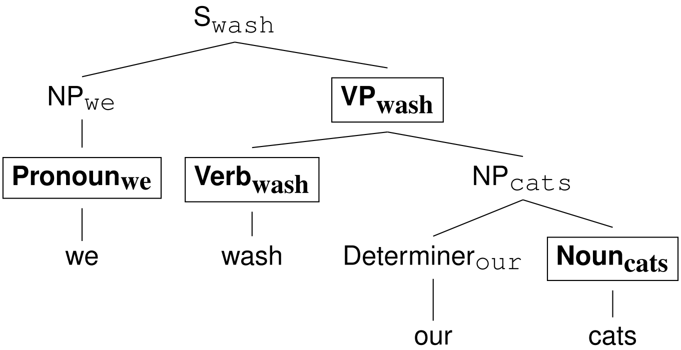
Phrase-structure tree

Example



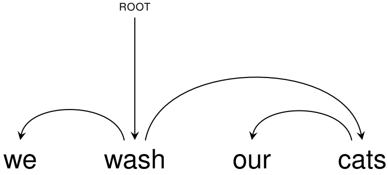
Phrase-structure tree with phrase-heads

Example



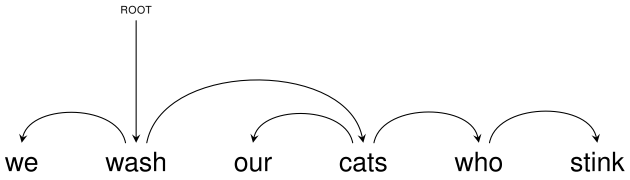
Phrase-structure tree with phrase-heads, **lexicalized**

Example

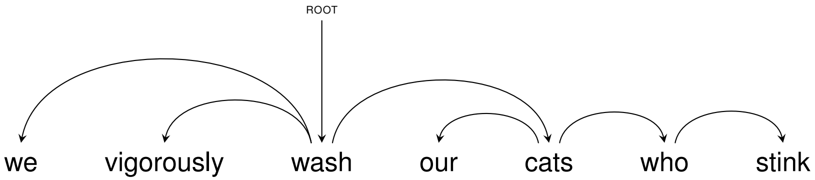


“Bare bones” dependency tree.

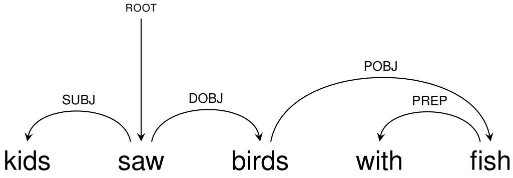
Example



Example



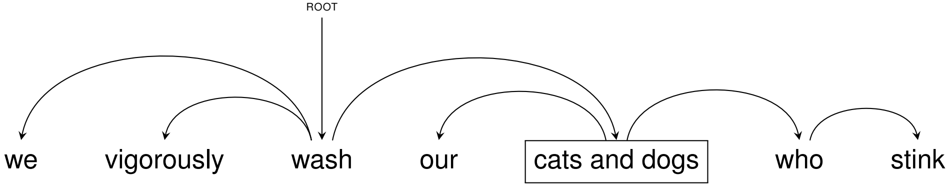
Labels



Key dependency relations captured in the labels include:

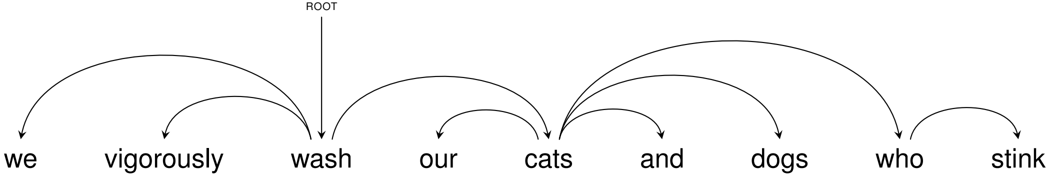
- Subject
- Direct Object
- Indirect Object
- Preposition Object
- Adjectival Modifier
- Adverbial Modifier

Problem: Coordination Structures



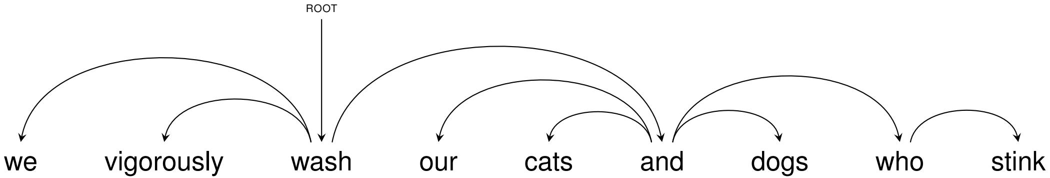
Most likely the most important problem with dependency syntax.

Coordination Structures: Proposal 1



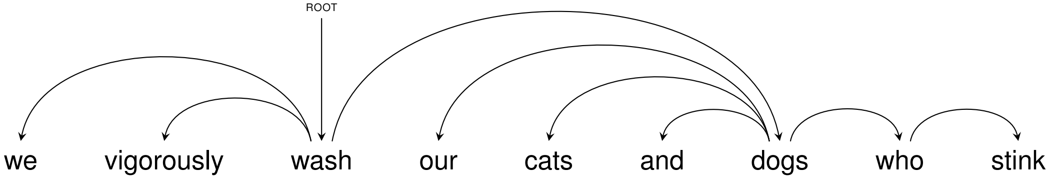
Make the first conjunct head?

Coordination Structures: Proposal 2



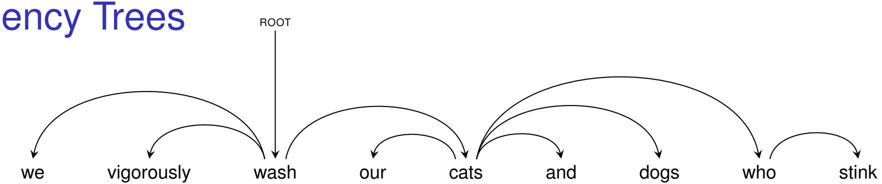
Make the coordinating conjunction the head?

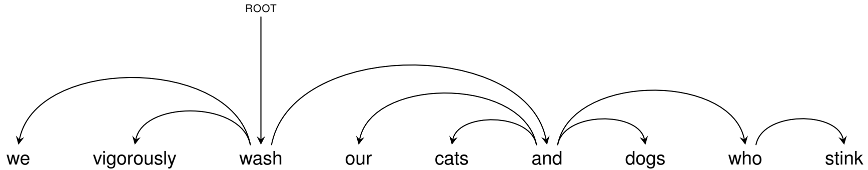
Coordination Structures: Proposal 3

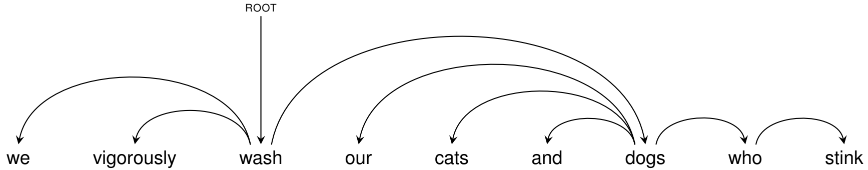


Make the second conjunct the head?

Dependency Trees

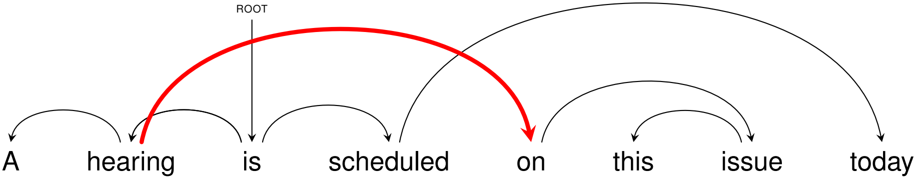






What is a common property among these trees?

Discontinuous Constituents / Crossing Arcs



Dependencies and Grammar

- Context-free grammars can be used to encode dependency structures.
- For every head word and group of its dependent children:

$$\mathrm{N}_{head} \rightarrow \mathrm{N}_{leftmost-sibling} \cdots \mathrm{N}_{head} \cdots \mathrm{N}_{rightmost-sibling}$$

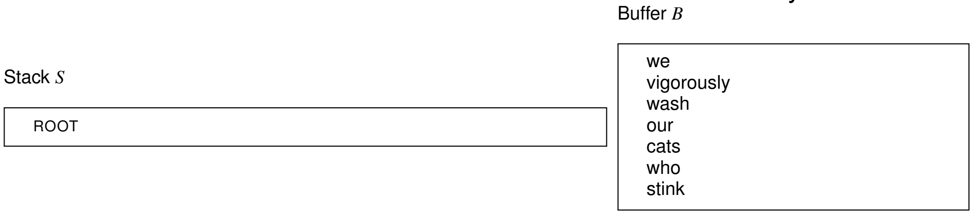
- And for every $c \in \mathcal{V}$: $\mathrm{N}_v \rightarrow v$ and $\mathrm{S} \rightarrow \mathrm{N}_v$
- Such a grammar can produce only **projective trees**, which are (informally) trees in which the arcs don't cross.

Three Approaches to Dependency Parsing

1. Dynamic Programming with bilexical dependency grammars
2. Transition-based parsing with a stack
3. Chu-Liu-Edmonds algorithm for the maximum spanning tree

Transition-based Parsing

- Process $\boldsymbol{x}$ once, from left to right, making a sequence of greedy parsing decisions.
- Formally, the parser is a **state machine** (*not* a finite-state machine) whose state is represented by a stack *S* and a buffer *B*.
- Initialize the buffer to contain $\boldsymbol{x}$ and the stack to contain the ROOT symbol.



Buffer B

Stack S

ROOT

we
vigorously
wash
our
cats
who
stink

- We can take one of three actions:
  - SHIFT the word at the front of the buffer B onto the stack S.
  - RIGHT-ARC: $u = \mathtt{pop}(S); v = \mathtt{pop}(S); \mathtt{push}(S, v \rightarrow u)$.
  - LEFT-ARC: $u = \mathtt{pop}(S); v = \mathtt{pop}(S); \mathtt{push}(S, v \leftarrow u)$.
    (for labeled parsing, add labels to the LEFT-ARC and RIGHT-ARC transitions.
- During parsing, apply a **classifier** to decide which transition to take next, greedily. No backtracking!

Transition-based Parsing Example

Buffer B

we
vigorously
wash
our
cats
who
stink

Stack S

ROOT

Actions:

Transition-based Parsing Example

Buffer B

vigorously
wash
our
cats
who
stink

Stack S

we
ROOT

Actions: SHIFT

Transition-based Parsing Example

Stack S

vigorously
we
ROOT

Buffer B

wash
our
cats
who
stink

Actions: SHIFT SHIFT

Transition-based Parsing Example

Stack S

wash
vigorously
we
ROOT

Buffer B

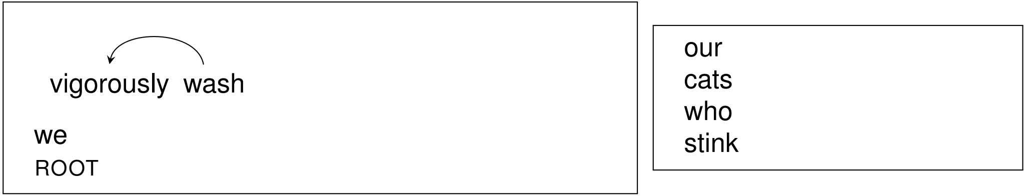
our
cats
who
stink

Actions: SHIFT SHIFT SHIFT

Transition-based Parsing Example

Stack S

Buffer B



vigorously wash

we

ROOT

our

cats

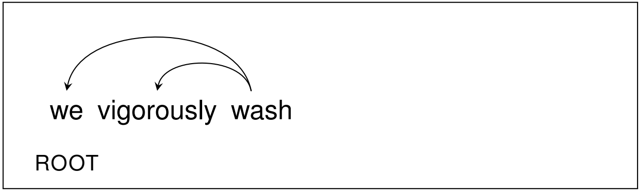
who

stink

Actions: SHIFT SHIFT SHIFT LEFT-ARC

Transition-based Parsing Example

Stack S



we vigorously wash

ROOT

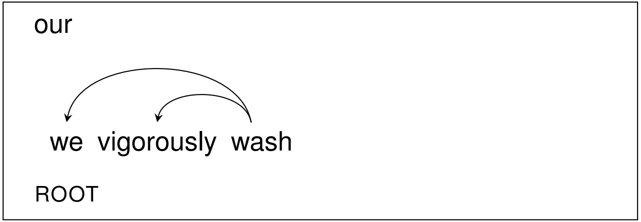
Buffer B

our
cats
who
stink

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC

Transition-based Parsing Example

Stack S



our

we vigorously wash

ROOT

Buffer B

cats
who
stink

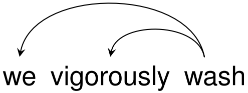
Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT

Transition-based Parsing Example

Stack S

cats

our



we vigorously wash

ROOT

Buffer B

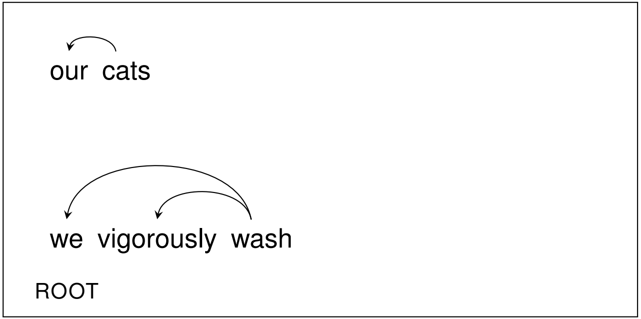
who

stink

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT

Transition-based Parsing Example

Stack S



our cats

we vigorously wash

ROOT

Buffer B

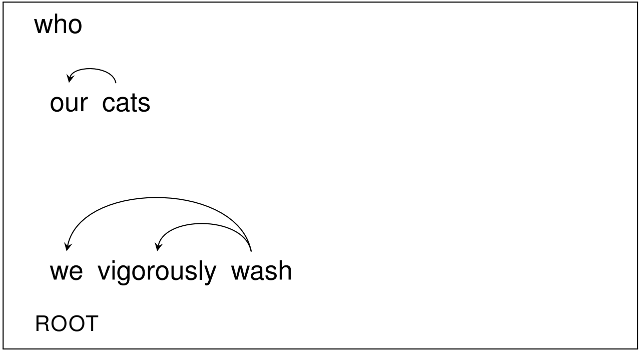
who

stink

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC

Transition-based Parsing Example

Stack S



who

our cats

we vigorously wash

ROOT

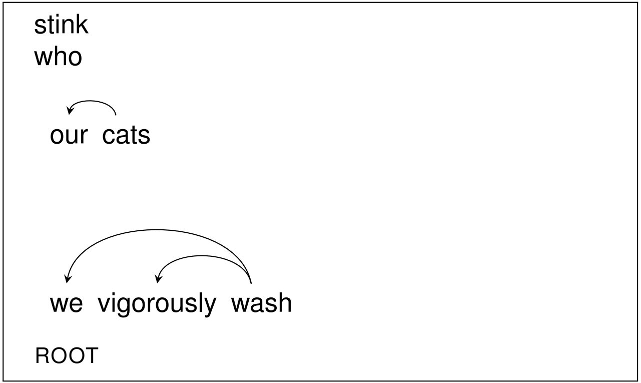
Buffer B

stink

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT

Transition-based Parsing Example

Stack S

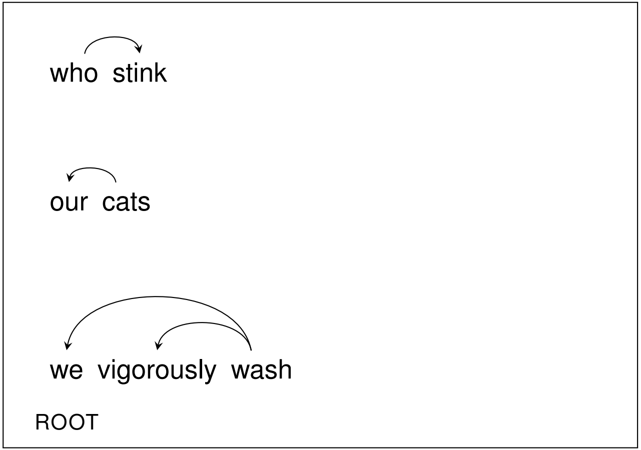


Buffer B

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT SHIFT

Transition-based Parsing Example

Stack S



who stink

our cats

we vigorously wash

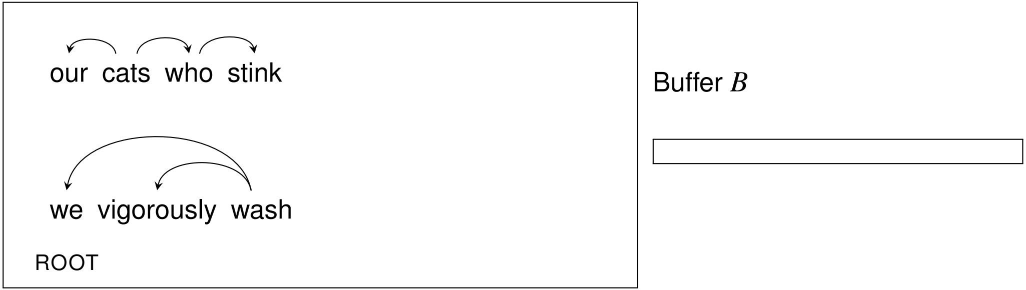
ROOT

Buffer B

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT SHIFT RIGHT-ARC

Transition-based Parsing Example

Stack S



our cats who stink

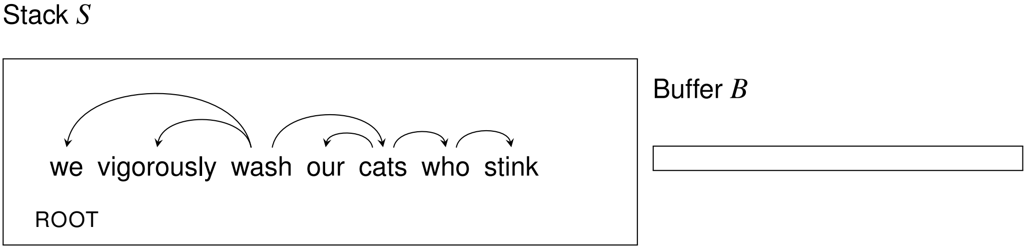
we vigorously wash

ROOT

Buffer B

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT SHIFT RIGHT-ARC RIGHT-ARC

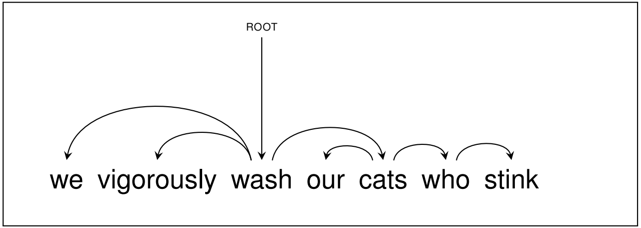
Transition-based Parsing Example



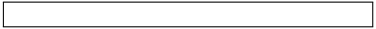
Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT SHIFT RIGHT-ARC RIGHT-ARC

Transition-based Parsing Example

Stack S



Buffer B



Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT SHIFT RIGHT-ARC RIGHT-ARC RIGHT-ARC

The Core of Transition-based Parsing

- At each iteration, choose among $\{$SHIFT, RIGHT-ARC, LEFT-ARC$\}$.
  - Actually, among all $\mathcal{L}$-labeled variants of RIGHT- and LEFT-ARC.
- Features can come from S, B, and the history of past actions – usually there is no decomposition into local structures.
- Training data: Dependency treebank trees converted into “oracle” transition sequence.
  - These transition sequences gives the right tree,
  - $2 \cdot n$ pairs: $\langle state, correcttransition \rangle$.
  - Each word gets SHIFTed once and participates as a child in one ARC.

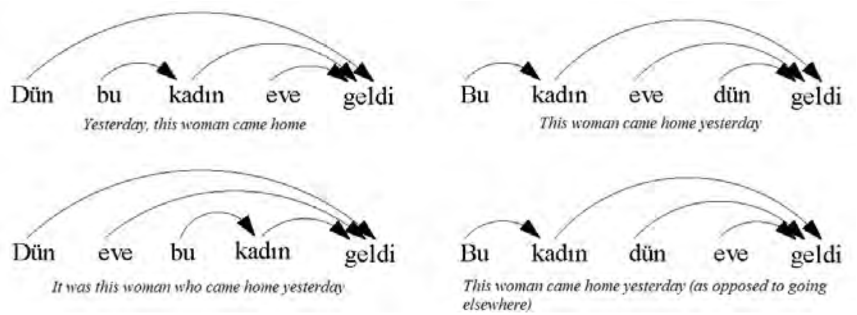
Transition-based Parsing: Remarks

- Can also be applied to phrase-structure parsing. Keyword: “shift-reduce” parsing.
- The algorithm for making decisions doesn’t need to be greedy; can maintain multiple hypotheses.
  - e.g., beam search
- Potential flaw: the classifier is typically trained under the assumption that previous classification decisions were all correct. As yet, no principled solution to this problem, but there are approximations based on “dynamic oracles”.

Dependency Parsing Evauation

- **Unlabeled attachment score**: Did you identify the head and the dependent correctly?
- **Labeled attachment score**: Did you identify the head and the dependent AND the label correctly?

Dependency Examples from Other Languages



Dün bu kadın eve geldi

Yesterday, this woman came home

Bu kadın eve dün geldi

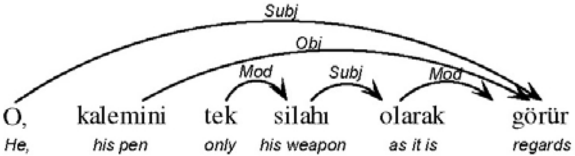
This woman came home yesterday

Dün eve bu kadın geldi

It was this woman who came home yesterday

Bu kadın dün eve geldi

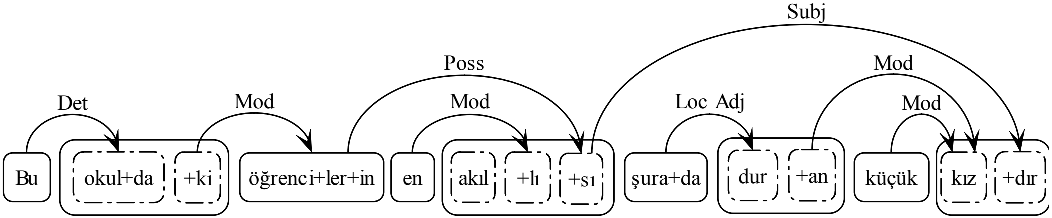
This woman came home yesterday (as opposed to going elsewhere)



O, kalemini tek silahı olarak görür

He, his pen only his weapon as it is regards

Dependency Examples from Other Languages



Bu	okul+da	+ki	öğrenci+ler+in	en	akıl	+lı	+sı	şura+da	dur	+an	küçük	kız	+dır
bu +Det	okul +Noun +A3sg +Pnon +Loc	+Adj	öğrenci +Noun +A3pl +Pnon +Gen	en +Adv	akıl +Noun +A3sg +Pnon +Nom	+Adj +With	+Noun +Zero +A3sg +P3sg +Nom	şura +Noun +A3sg +Pnon +Loc	dur +Verb +Pos	+Adj +Prespart	küçük +Adj	kız +Noun +A3sg +Pnon +Nom	+Verb +Zero +Pres +Cop +A3s
This	school-at+that-is		student-s-'	most	intelligence+with+of			there	stand+ing		little	girl+is	

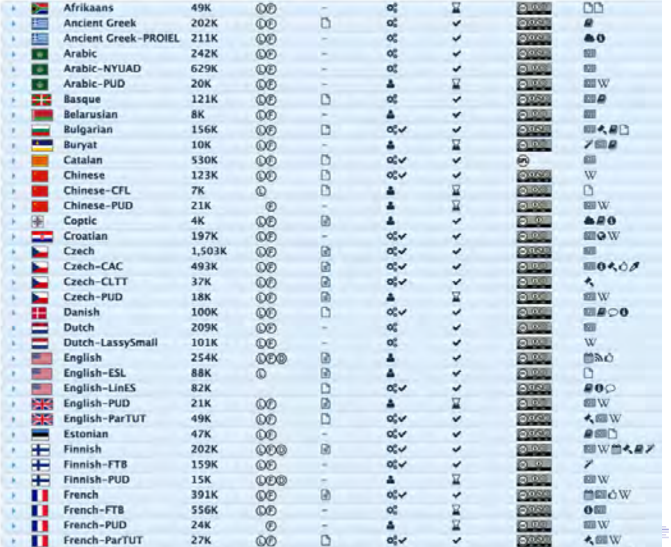
The most intelligent of the students in this school is the little girl standing there.

Dependency Examples from Other Languages

```
<S>
<W IX=1 LEM="bu" MORPH="bu" IG=[(1, "bu+Det")] REL=[(3,1,(DETERMINER)]>
Bu </W>
<W IX=2 LEM="eski"’ MORPH="eski" IG=[(1, "eski+Adj")]
REL=[3,1,(MODIFIER)]> eski> </W>
<W IX=3 LEM="bahçe" MORPH="bahçe+DA+ki" IG=[(1, "bahçe+A3sg+Pnon+Loc")
(2, "+Adj+Rel")] REL=[4,1,(MODIFIER)]> bahçedeki </W>
<W IX=4 LEM="gül" MORPH="gül+nHn" IG=[(1,"gül+Noun+A3sg+Pnon+Gen")]
REL=[6,1,(SUBJECT)]> gülün </W>
<W IX=5 LEM="böyle" MORPH="böyle" IG=[(1,"böyle+Adv")]
REL=[6,1,(MODIFIER)]> böyle </W>
<W IX=6 LEM="büyü" MORPH="büyü+mA+sH" IG=[(1,"büyü+Verb+Pos") (2,
"+Noun+Inf+A3sg+P3sg+Nom")] REL=[9,1,(SUBJECT)]> büyümesi </W>
<W IX=7 LEM="herkes" MORPH="herkes+yH"
IG=[(1,"herkes+Pron+A3sg+Pnon+Acc")] REL=[9,1,(OBJECT)]> herkesi </W>
<W IX=8 LEM="çok" MORPH="çok" IG=[(1,"çok+Adv’’)] REL=[9,1,(MODIFIER)]>
çok </W>
<W IX=9 LEM="etkile" MORPH="etkile+DH" IG=[(1,
"etkile+Verb+Pos+Past+A3sg")] REL=[]> etkiledi </W>
</S>
```

Universal Dependencies

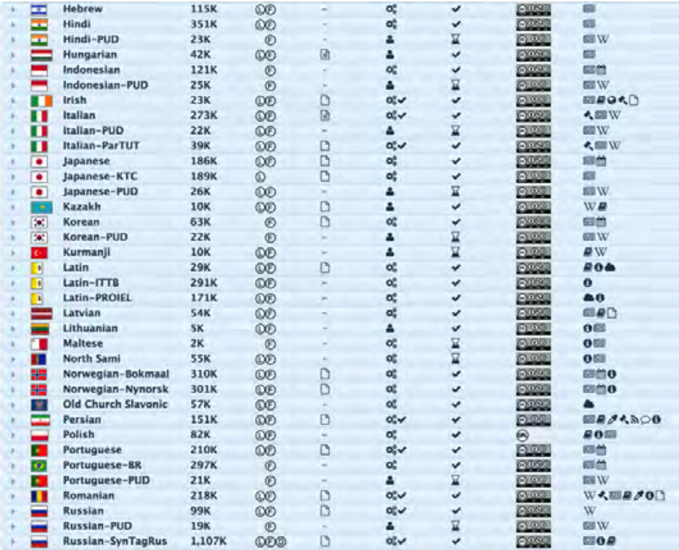
- A very recent project that aims to use a small set of “universal” labels and annotation guidelines (`universaldependencies.org`).



Afrikaans	49K	–	⌛	
Ancient Greek	202K	🗋	✓	
Ancient Greek-PROIEL	211K	–	✓	
Arabic	242K	–	✓	
Arabic-NYUAD	629K	–	✓	
Arabic-PUD	20K	–	⌛	W
Basque	121K	🗋	✓	
Belarusian	8K	–	✓	
Bulgarian	156K	🗋	✓	
Buryat	10K	–	⌛	
Catalan	530K	🗋	✓	
Chinese	123K	🗋	✓	W
Chinese-CFL	7K	🗋	⌛	
Chinese-PUD	21K	–	⌛	W
Coptic	4K	🗋	✓	
Croatian	197K	–	✓	W
Czech	1,503K	🗋	✓	
Czech-CAC	493K	🗋	✓	
Czech-CLTT	37K	🗋	✓	
Czech-PUD	18K	🗋	⌛	W
Danish	100K	🗋	✓	
Dutch	209K	–	✓	
Dutch-LassySmall	101K	–	✓	W
English	254K	🗋	✓	
English-ESL	88K	🗋	✓	
English-LinES	82K	🗋	✓	
English-PUD	21K	🗋	⌛	W
English-ParTUT	49K	🗋	✓	W
Estonian	47K	–	✓	
Finnish	202K	🗋	✓	W
Finnish-FTB	159K	–	✓	
Finnish-PUD	15K	–	⌛	W
French	391K	🗋	✓	W
French-FTB	556K	–	⌛	
French-PUD	24K	–	⌛	W
French-ParTUT	27K	🗋	✓	W

Universal Dependencies

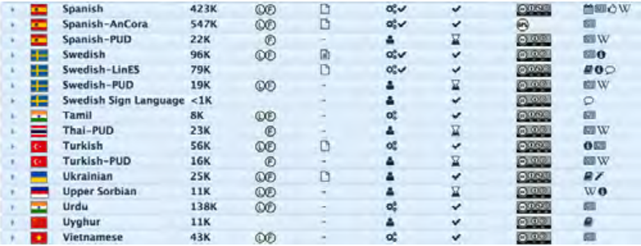
- A very recent project that aims to use a small set of “universal” labels and annotation guidelines (`universaldependencies.org`).



Hebrew	115K	ⓁⒻ	-	⚙	✓	🅭	📰
Hindi	351K	ⓁⒻ	-	⚙	✓	🅭	📰
Hindi-PUD	23K	Ⓕ	-	👤	⧗	🅭	📰W
Hungarian	42K	ⓁⒻ	📄	👤	✓	🅭	📰
Indonesian	121K	Ⓕ	-	⚙	✓	🅭	📰📅
Indonesian-PUD	25K	Ⓕ	-	👤	⧗	🅭	📰W
Irish	23K	ⓁⒻ	🗋	⚙✓	✓	🅭	📰📖🌐🔨🗋
Italian	273K	ⓁⒻ	📄	⚙✓	✓	🅭	🔨📰W
Italian-PUD	22K	ⓁⒻ	-	👤	⧗	🅭	📰W
Italian-ParTUT	39K	ⓁⒻ	🗋	⚙✓	✓	🅭	🔨📰W
Japanese	186K	ⓁⒻ	🗋	⚙	✓	🅭	📰📅
Japanese-KTC	189K	Ⓛ	🗋	⚙	✓	🅭	📰
Japanese-PUD	26K	ⓁⒻ	-	👤	⧗	🅭	📰W
Kazakh	10K	ⓁⒻ	🗋	👤	✓	🅭	W📖
Korean	63K	Ⓕ	🗋	⚙	✓	🅭	📰📅
Korean-PUD	22K	Ⓕ	-	👤	⧗	🅭	📰W
Kurmanji	10K	ⓁⒻ	-	👤	⧗	🅭	📖W
Latin	29K	ⓁⒻ	🗋	⚙	✓	🅭	📖ℹ☁
Latin-ITTB	291K	ⓁⒻ	-	⚙	✓	🅭	ℹ
Latin-PROIEL	171K	ⓁⒻ	-	⚙	✓	🅭	☁ℹ
Latvian	54K	ⓁⒻ	-	⚙	✓	🅭	📰📖🗋
Lithuanian	5K	ⓁⒻ	-	👤	✓	🅭	ℹ📰
Maltese	2K	Ⓕ	-	⚙	⧗	🅭	ℹ📰
North Sami	55K	ⓁⒻ	-	⚙	⧗	🅭	ℹ📰
Norwegian-Bokmaal	310K	ⓁⒻ	🗋	⚙	✓	🅭	📰📅ℹ
Norwegian-Nynorsk	301K	ⓁⒻ	🗋	⚙	✓	🅭	📰📅ℹ
Old Church Slavonic	57K	ⓁⒻ	-	⚙	✓	🅭	☁
Persian	151K	ⓁⒻ	🗋	⚙✓	✓	🅭	📰📖✒🔨📶💬ℹ
Polish	82K	ⓁⒻ	-	⚙	✓	◎	📖ℹ📰
Portuguese	210K	ⓁⒻ	🗋	⚙✓	✓	🅭	📰📅
Portuguese-BR	297K	Ⓕ	-	⚙	✓	🅭	📰📅
Portuguese-PUD	21K	Ⓕ	-	👤	⧗	🅭	📰W
Romanian	218K	ⓁⒻ	🗋	⚙✓	✓	🅭	W🔨📰📖✒ℹ🗋
Russian	99K	ⓁⒻ	🗋	⚙✓	✓	🅭	W
Russian-PUD	19K	Ⓕ	-	👤	⧗	🅭	📰W
Russian-SynTagRus	1,107K	ⓁⒻⒹ	🗋	⚙✓	✓	🅭	📰ℹ📖

Universal Dependencies

- A very recent project that aims to use a small set of "universal" labels and annotation guidelines (`universaldependencies.org`).



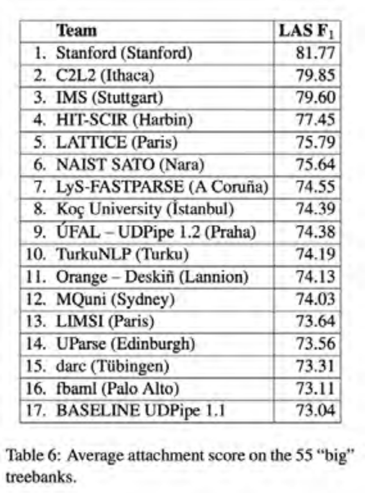
Spanish	423K	ⓁⒻ	🗋	⚙✓	✓	W
Spanish-AnCora	547K	ⓁⒻ	🗋	⚙✓	✓	
Spanish-PUD	22K	Ⓕ	-	👤	⧗	W
Swedish	96K	ⓁⒻ	🗋	⚙✓	✓	
Swedish-LinES	79K		🗋	⚙✓	✓	
Swedish-PUD	19K	ⓁⒻ	-	👤	⧗	W
Swedish Sign Language	<1K		-	👤	✓	
Tamil	8K	ⓁⒻ	-	⚙	✓	
Thai-PUD	23K	Ⓕ	-	👤	⧗	W
Turkish	56K	ⓁⒻ	🗋	⚙	✓	
Turkish-PUD	16K	Ⓕ	-	👤	⧗	W
Ukrainian	25K	ⓁⒻ	🗋	👤	✓	
Upper Sorbian	11K	ⓁⒻ	-	👤	⧗	W
Urdu	138K	ⓁⒻ	-	⚙	✓	
Uyghur	11K		-	👤	✓	
Vietnamese	43K	ⓁⒻ	-	⚙	✓	

State-of-the-art Dependency Parsers

- Stanford Parser
  - Detailed Information at `https://nlp.stanford.edu/software/lex-parser.shtml`
  - Demo at `http://nlp.stanford.edu:8080/parser/`
- MaltParser is the original transition-based dependency parser by Nivre.
  - “MaltParser is a system for data-driven dependency parsing, which can be used to induce a parsing model from treebank data and to parse new data using an induced model.”
  - Available at `http://maltparser.org/`

State-of-the-art Dependency Parser Performance

CONLL Shared Task Results

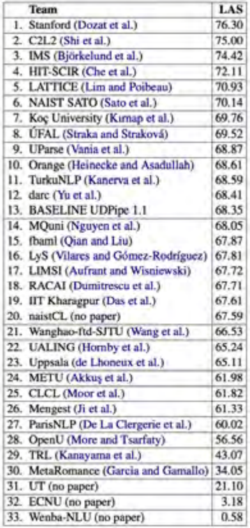


Team	LAS F_1
1. Stanford (Stanford)	81.77
2. C2L2 (Ithaca)	79.85
3. IMS (Stuttgart)	79.60
4. HIT-SCIR (Harbin)	77.45
5. LATTICE (Paris)	75.79
6. NAIST SATO (Nara)	75.64
7. LyS-FASTPARSE (A Coruña)	74.55
8. Koç University (İstanbul)	74.39
9. ÚFAL – UDPipe 1.2 (Praha)	74.38
10. TurkuNLP (Turku)	74.19
11. Orange – Deskiñ (Lannion)	74.13
12. MQuni (Sydney)	74.03
13. LIMSI (Paris)	73.64
14. UParse (Edinburgh)	73.56
15. darc (Tübingen)	73.31
16. fbaml (Palo Alto)	73.11
17. BASELINE UDPipe 1.1	73.04

Table 6: Average attachment score on the 55 "big" treebanks.

State-of-the-art Dependency Parser Performance

CONLL Shared Task Results

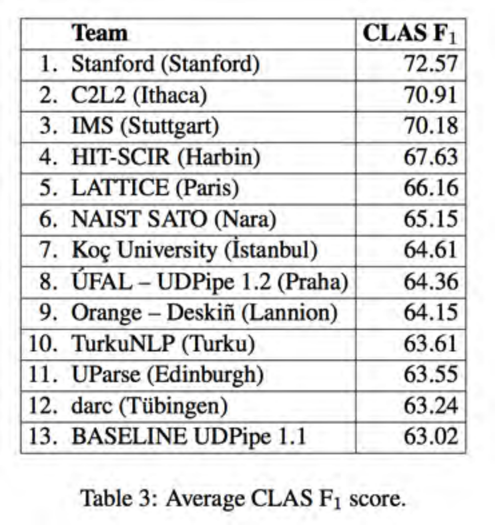


	Team	LAS
1.	Stanford (Dozat et al.)	76.30
2.	C2L2 (Shi et al.)	75.00
3.	IMS (Björkelund et al.)	74.42
4.	HIT-SCIR (Che et al.)	72.11
5.	LATTICE (Lim and Poibeau)	70.93
6.	NAIST SATO (Sato et al.)	70.14
7.	Koç University (Kırnap et al.)	69.76
8.	ÚFAL (Straka and Straková)	69.52
9.	UParse (Vania et al.)	68.87
10.	Orange (Heinecke and Asadullah)	68.61
11.	TurkuNLP (Kanerva et al.)	68.59
12.	darc (Yu et al.)	68.41
13.	BASELINE UDPipe 1.1	68.35
14.	MQuni (Nguyen et al.)	68.05
15.	fbaml (Qian and Liu)	67.87
16.	LyS (Vilares and Gómez-Rodríguez)	67.81
17.	LIMSI (Aufrant and Wisniewski)	67.72
18.	RACAI (Dumitrescu et al.)	67.71
19.	IIT Kharagpur (Das et al.)	67.61
20.	naistCL (no paper)	67.59
21.	Wanghao-ftd-SJTU (Wang et al.)	66.53
22.	UALING (Hornby et al.)	65.24
23.	Uppsala (de Lhoneux et al.)	65.11
24.	METU (Akkuş et al.)	61.98
25.	CLCL (Moor et al.)	61.82
26.	Mengest (Ji et al.)	61.33
27.	ParisNLP (De La Clergerie et al.)	60.02
28.	OpenU (More and Tsarfaty)	56.56
29.	TRL (Kanayama et al.)	43.07
30.	MetaRomance (Garcia and Gamallo)	34.05
31.	UT (no paper)	21.10
32.	ECNU (no paper)	3.18
33.	Wenba-NLU (no paper)	0.58

Table 2: Ranking of the participating systems by the main evaluation metric, the labeled attachment F_1-score, macro-averaged over 81 test sets.

State-of-the-art Dependency Parser Performance

CONLL Shared Task Results



	Team	CLAS F_1
1.	Stanford (Stanford)	72.57
2.	C2L2 (Ithaca)	70.91
3.	IMS (Stuttgart)	70.18
4.	HIT-SCIR (Harbin)	67.63
5.	LATTICE (Paris)	66.16
6.	NAIST SATO (Nara)	65.15
7.	Koç University (İstanbul)	64.61
8.	ÚFAL – UDPipe 1.2 (Praha)	64.36
9.	Orange – Deskiñ (Lannion)	64.15
10.	TurkuNLP (Turku)	63.61
11.	UParse (Edinburgh)	63.55
12.	darc (Tübingen)	63.24
13.	BASELINE UDPipe 1.1	63.02

Table 3: Average CLAS F_1 score.

11-411
Natural Language Processing
Lexical Semantics

Kemal Oflazer

Carnegie Mellon University in Qatar

Lexical Semantics

The study of meanings of words:

- Decompositional
  - Words have component meanings
  - Total meanings are composed of these component meanings
- Ontological
  - The meanings of words can be defined in relation to other words.
  - Paradigmatic – Thesaurus-based
- Distributional
  - The meanings of words can be defined in relation to their contexts among other words.
  - Syntagmatic – meaning defined by syntactic context

Decompositional Lexical Semantics

- Assume that *woman* has (semantic) components [female], [human], and [adult].
- *Man* might have the componets [male], [human], and [adult].
- Such “semantic features” can be combined to form more complicated meanings.
- Although this looks appealing, there is a little bit of a chickens-and-eggs situation.
- Scholars and language scientists have not yet developed a consensus about a common set of “semantic primitives.”
- Such as representation probably has to involve more structure than just a flat set of features per word.

Ontological Semantics

Relations between words/senses

- Synonymy
- Antonymy
- Hyponymy/Hypernymy
- Meronymy/Holonymy

Key resource: Wordnet

Terminology: Lemma and Wordform

- A **lemma** or **citation form**
  - Common stem, part of speech, rough semantics
- A **wordform**
  - The (inflected) word as it appears in text

Wordform	**Lemma**
banks	bank
sung	sing
sang	sing
went	go
goes	go

Lemmas have Senses

- One lemma “bank” can have many meanings:
  - Sense 1: “. . . a bank$_1$ can hold the investments in a custodial account . . . ”
  - Sense 2: “. . . as agriculture burgeons on the east bank$_2$ the river will shrink even more.”
- **Sense** (or **word sense**)
  - A discrete representation of an aspect of a word’s meaning.
- The lemma **bank** here has two senses.

Homonymy

- **Homonyms**: words that share a form but have unrelated, distinct meanings:
  - bank$_1$: financial institution, bank$_2$: sloping land
  - bat$_1$: club for hitting a ball, bat$_2$: nocturnal flying mammal
- **Homographs**: Same spelling (bank/bank, bat/bat)
- **Homophones**: Same pronunciation
  - write and right
  - piece and peace

Homonymy causes problems for NLP applications

- Information retrieval
  - “bat care”
- Machine Translation
  - bat: murciélago (animal) or bate (for baseball)
- Text-to-Speech
  - bass (stringed instrument) vs. bass (fish)

Polysemy

- The bank$_1$ was constructed in 1875 out of local red brick.
- I withdrew the money from the bank$_2$.
- Are those the same sense?
  - bank$_1$: “The building belonging to a financial institution”
  - bank$_2$: “A financial institution”
- A **polysemous** word has multiple related meanings.
- Most non-rare words have multiple meanings

Metonymy/Systematic Polysemy

- Lots of types of polysemy are systematic
  - *school*, *university*, *hospital*
  - All can mean the institution or the building.
- A systematic relation
  - Building ⇔ Organization
- Other examples of such polysemy
  - Author (Jane Austen wrote Emma) ⇔ Works of Author (I love Jane Austen)
  - Tree (Plums have beautiful blossoms) ⇔ Fruit (I ate a preserved plum)

How do we know when a word has multiple senses?

- The “zeugma”[1] test: Two senses of serve?
  - Which flights serve breakfast?
  - Does Qatar Airways serve Philadelphia?
  - ?Does Qatar Airways serve breakfast and Washington?
- Since this conjunction sounds weird, we say that these are two different senses of “serve”
- “The farmers in the valley grew potatoes, peanuts, and bored.”
- “He lost his coat and his temper.”

[1] A zeugma is an interesting device that can cause confusion in sentences, while also adding some flavor.

Synonymy

- Words *a* and *b* share an identical sense or have the same meaning in some or all contexts.
  - filbert / hazelnut
  - couch / sofa
  - big / large
  - automobile / car
  - vomit / throw up
  - water / H_2O
- Synonyms can be substituted for each other in all situations.
- True synonymy is relatively rare compared to other lexical relations.
  - may not preserve the acceptability based on notions of politeness, slang, register, genre, etc.
  - water / H_2O
  - big / large
  - bravery / courage
    - Bravery is the ability to confront pain, danger, or attempts of intimidation without any feeling of fear.
    - Courage, on the other hand, is the ability to undertake an overwhelming difficulty or pain despite the eminent and unavoidable presence of fear.

Synonymy

Synonymy is a relation between senses rather than words.

- Consider the words *big* and *large*. Are they synonyms?
  - How **big** is that plane?
  - Would I be flying on a **large** or small plane?
- How about here:
  - Miss Nelson became a kind of **big** sister to Benjamin.
  - ?Miss Nelson became a kind of **large** sister to Benjamin.
- Why?
  - *big* has a sense that means being older, or grown up
  - *large* lacks this sense.

Antonymy

- Lexical items *a* and *b* have senses which are “opposite”, with respect to one feature of meaning
- Otherwise they are similar
    - dark/light
    - short/long
    - fast/slow
    - rise/fall
    - hot/cold
    - up/down
    - in/out
- More formally: antonyms can
    - define a binary opposition or be at opposite ends of a scale (long/short, fast/slow)
    - or be **reversives** (rise/fall, up/down)
- Antonymy is much more common than true synonymy.
- Antonymy is not always well defined, especially for nouns (but for other words as well).

Hyponymy/Hypernymy

- The “is-a” relations.
- Lexical item *a* is a **hyponym** of lexical item *b* if *a* is a kind of *b* (if a sense of *b* refers to a superset of the referent of a sense of *a*).
- *screwdriver* is a hyponym of *tool*.
- *screwdriver* is also a hyponym of *drink*.
- *car* is a hyponym of *vehicle*
- *mango* is a hyponym of *fruit*
- **Hypernymy** is the converse of hyponymy.
- *tool* and *drink* are hypernyms of screwdriver.
- *vehicle* is a hypernym of *car*

Hyponymy more formally

- Extensional
  - The class denoted by the superordinate (e.g., vehicle) extensionally includes the class denoted by the hyponym (e.g. car).
- Entailment
  - A sense A is a hyponym of sense B if being an A entails being a B (e.g. if it is car, it is a vehicle)
- Hyponymy is usually transitive
  - If A is a hyponym of B and B is a hyponym of $C \Rightarrow A$ is a hyponym of C.
- Another name is the IS-A hierarchy
  - A IS-A B
  - B **subsumes** A

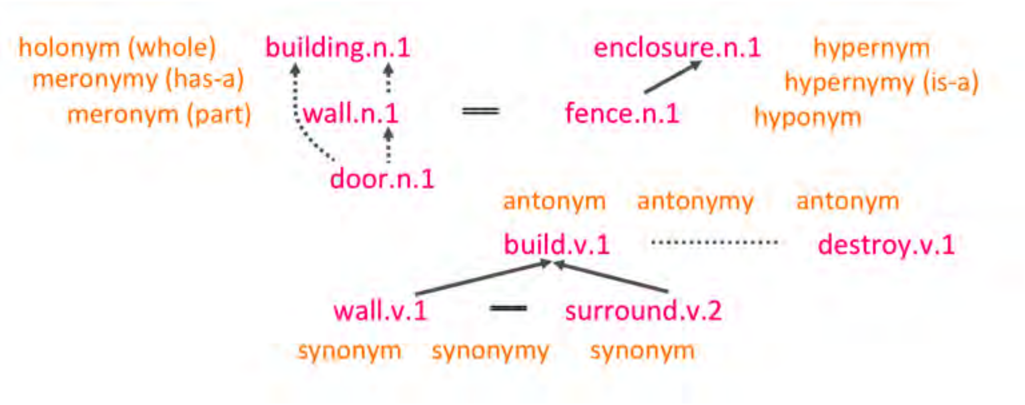
Hyponyms and Instances

- An **instance** is an individual, a proper noun that is a unique entity
  - Doha/San Francisco/London are instances of *city*.
- But *city* is a class
  - *city* is a hyponym of *municipality* . . . *location* . . .

Meronymy/Holonymy

- The “part-of” relation
- Lexical item *a* is a meronym of lexical item *b* if a sense of *a* is a part of/a member of a sense of *b*.
- *hand* is a meronym of *body*.
- *congressperson* is a meronym of *congress*.
- *Holonymy* (think: whole) is the converse of *meronymy*.
- *body* is a holonym of *hand*.

A Lexical Mini-ontology



WordNet

- A hierarchically organizated database of (English) word senses.
- George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.
- Available at `wordnet.princeton.edu`
- Provides a set of three lexical databases:
  - Nouns
  - Verbs
  - Adjectives and adverbs.
- Relations are between senses, not lexical items (words).
- Applications Program Interfaces (APIs) are available for many languages and toolkits including a Python interface via NLTK.
- WordNet 3.0

Category	**Unique Strings**
Noun	117,197
Verb	11,529
Adjective	22,429
Adverb	4,481

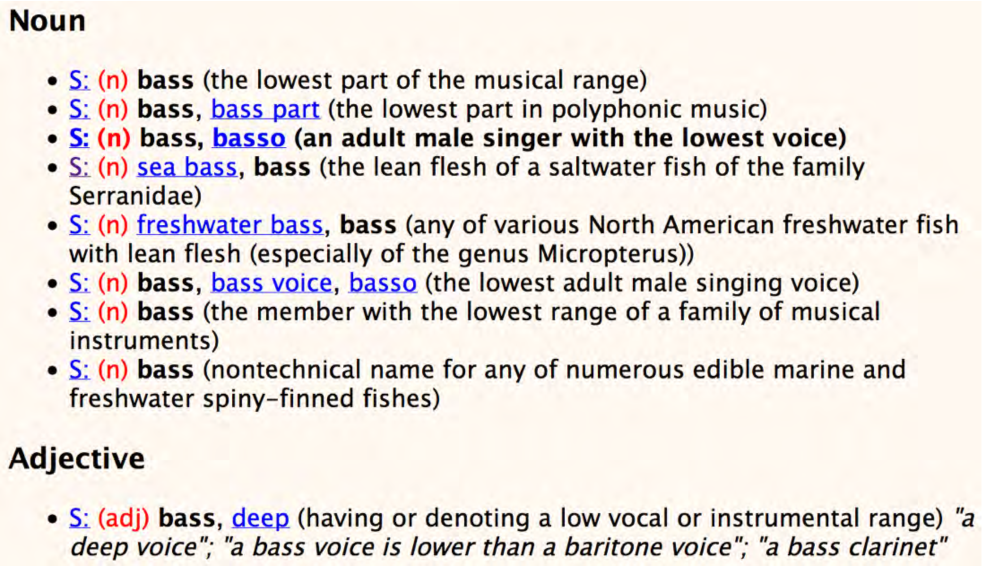
Synsets

- Primitive in WordNet: Synsets (roughly: synonym sets)
  - Words that can be given the same gloss or definition.
  - Does not require absolute synonymy
- For example: {*chump*, *fool*, *gull*, *mark*, *patsy*, *fall guy*, *sucker*, *soft touch*, *mug*}
- Other lexical relations, like *antonymy*, *hyponymy*, and *meronymy* are between synsets, not between senses directly.

Synsets for *dog (n)*

- S: (n) **dog, domestic dog, Canis familiaris** (a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) “the dog barked all night”
- S: (n) **frump, dog** (a dull unattractive unpleasant girl or woman) “she got a reputation as a frump”, “she’s a real dog”
- S: (n) **dog** (informal term for a man) “you lucky dog”
- S: (n) **cad, bounder, blackguard, dog, hound, heel** (someone who is morally reprehensible) “you dirty dog”
- S: (n) **frank, frankfurter, hotdog, hot dog, dog, wiener, wienerwurst, weenie** (a smooth-textured sausage of minced beef or pork usually smoked; olen served on a bread roll)
- S: (n) **pawl, detent, click, dog** (a hinged catch that fits into a notch of a ratchet to move a wheel forward or prevent it from moving backward)
- S: (n) **andiron, firedog, dog, dog-iron** (metal supports for logs in a fireplace) “the andirons were too hot to touch”

Synsets for *bass* in WordNet



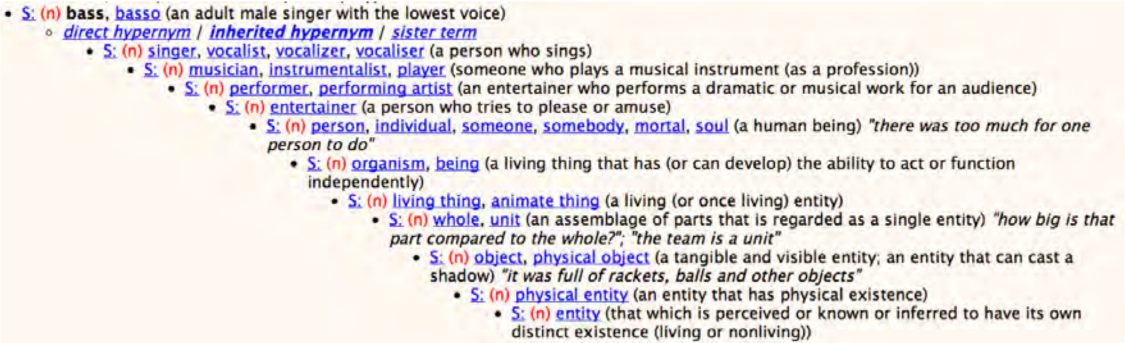
Noun

- S: (n) **bass** (the lowest part of the musical range)
- S: (n) **bass**, bass part (the lowest part in polyphonic music)
- **S: (n) bass, basso (an adult male singer with the lowest voice)**
- S: (n) sea bass, **bass** (the lean flesh of a saltwater fish of the family Serranidae)
- S: (n) freshwater bass, **bass** (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
- S: (n) **bass**, bass voice, basso (the lowest adult male singing voice)
- S: (n) **bass** (the member with the lowest range of a family of musical instruments)
- S: (n) **bass** (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

Adjective

- S: (adj) **bass**, deep (having or denoting a low vocal or instrumental range) *"a deep voice"; "a bass voice is lower than a baritone voice"; "a bass clarinet"*

Hierarchy for *bass*$_3$ in WordNet

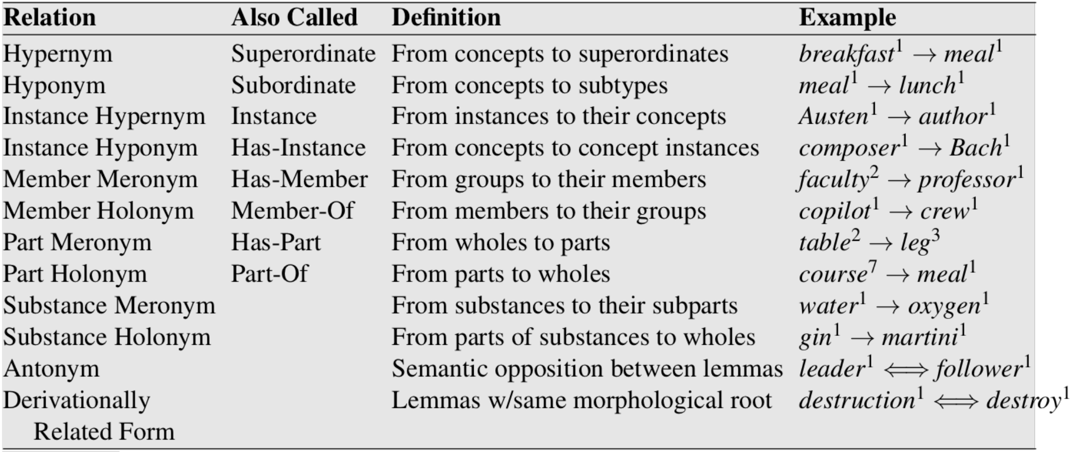


- S: (n) **bass**, basso (an adult male singer with the lowest voice)
  - *direct hypernym* / ***inherited hypernym*** / *sister term*
    - S: (n) singer, vocalist, vocalizer, vocaliser (a person who sings)
      - S: (n) musician, instrumentalist, player (someone who plays a musical instrument (as a profession))
        - S: (n) performer, performing artist (an entertainer who performs a dramatic or musical work for an audience)
          - S: (n) entertainer (a person who tries to please or amuse)
            - S: (n) person, individual, someone, somebody, mortal, soul (a human being) *"there was too much for one person to do"*
              - S: (n) organism, being (a living thing that has (or can develop) the ability to act or function independently)
                - S: (n) living thing, animate thing (a living (or once living) entity)
                  - S: (n) whole, unit (an assemblage of parts that is regarded as a single entity) *"how big is that part compared to the whole?"; "the team is a unit"*
                    - S: (n) object, physical object (a tangible and visible entity; an entity that can cast a shadow) *"it was full of rackets, balls and other objects"*
                      - S: (n) physical entity (an entity that has physical existence)
                        - S: (n) entity (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

The IS-A Hierarchy for *fish (n)*

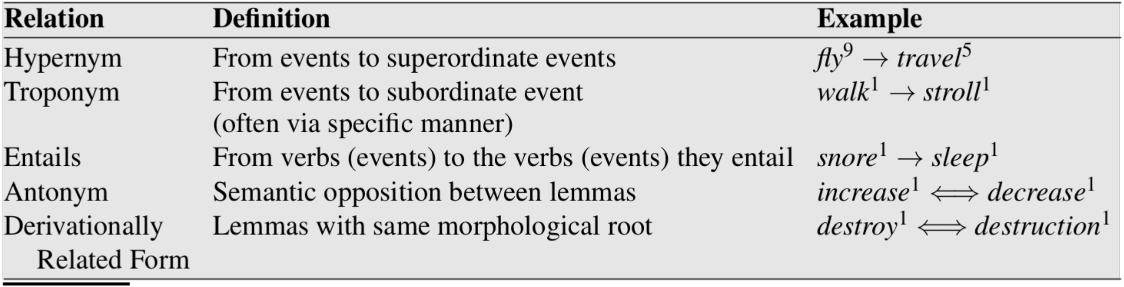
- fish (any of various mostly cold-blooded aquatic vertebrates usually having scales and breathing through gills)
- aquatic vertebrate (animal living wholly or chiefly in or on water)
- vertebrate, craniate (animals having a bony or cartilaginous skeleton with a segmented spinal column and a large brain enclosed in a skull or cranium)
- chordate (any animal of the phylum Chordata having a notochord or spinal column)
- animal, animate being, beast, brute, creature, fauna (a living organism characterized by voluntary movement)
- organism, being (a living thing that has (or can develop) the ability to act or function independently)
- living thing, animate thing (a living (or once living) entity)
- whole, unit (an assemblage of parts that is regarded as a single entity)
- object, physical object (a tangible and visible entity; an entity that can cast a shadow)
- entity (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

WordNet Noun Relations



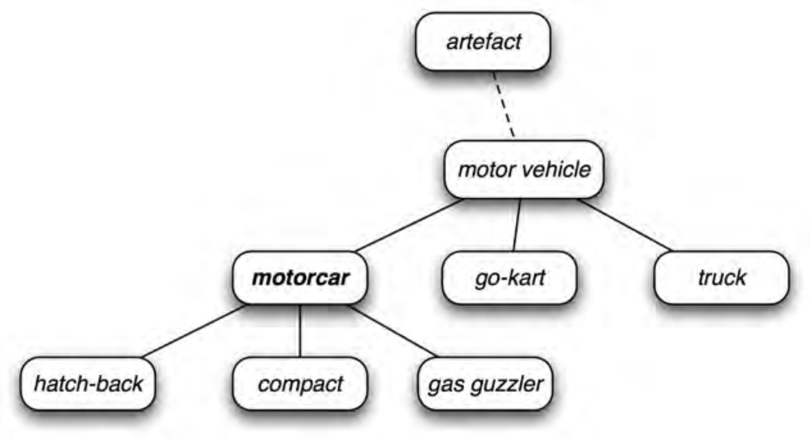
Relation	Also Called	Definition	Example
Hypernym	Superordinate	From concepts to superordinates	$\textit{breakfast}^1 \rightarrow \textit{meal}^1$
Hyponym	Subordinate	From concepts to subtypes	$\textit{meal}^1 \rightarrow \textit{lunch}^1$
Instance Hypernym	Instance	From instances to their concepts	$\textit{Austen}^1 \rightarrow \textit{author}^1$
Instance Hyponym	Has-Instance	From concepts to concept instances	$\textit{composer}^1 \rightarrow \textit{Bach}^1$
Member Meronym	Has-Member	From groups to their members	$\textit{faculty}^2 \rightarrow \textit{professor}^1$
Member Holonym	Member-Of	From members to their groups	$\textit{copilot}^1 \rightarrow \textit{crew}^1$
Part Meronym	Has-Part	From wholes to parts	$\textit{table}^2 \rightarrow \textit{leg}^3$
Part Holonym	Part-Of	From parts to wholes	$\textit{course}^7 \rightarrow \textit{meal}^1$
Substance Meronym		From substances to their subparts	$\textit{water}^1 \rightarrow \textit{oxygen}^1$
Substance Holonym		From parts of substances to wholes	$\textit{gin}^1 \rightarrow \textit{martini}^1$
Antonym		Semantic opposition between lemmas	$\textit{leader}^1 \Longleftrightarrow \textit{follower}^1$
Derivationally Related Form		Lemmas w/same morphological root	$\textit{destruction}^1 \Longleftrightarrow \textit{destroy}^1$

WordNet Verb Relations

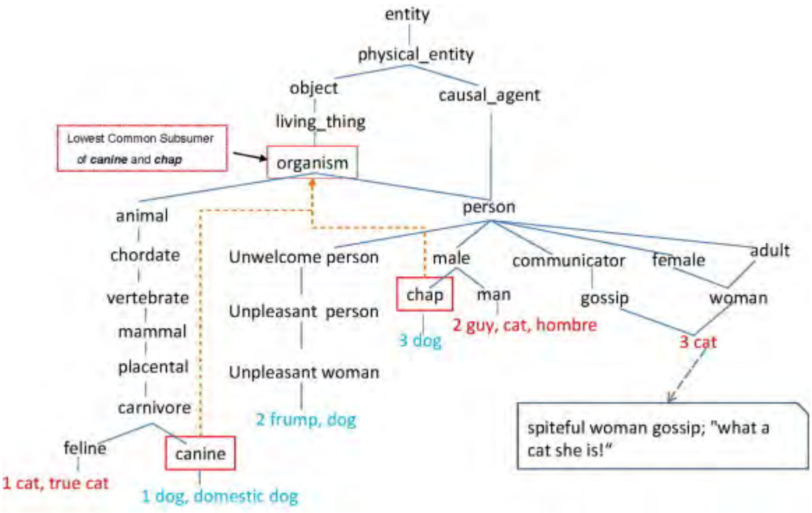


Relation	Definition	Example
Hypernym	From events to superordinate events	*fly*^{9} → *travel*^{5}
Troponym	From events to subordinate event (often via specific manner)	*walk*^{1} → *stroll*^{1}
Entails	From verbs (events) to the verbs (events) they entail	*snore*^{1} → *sleep*^{1}
Antonym	Semantic opposition between lemmas	*increase*^{1} ⟺ *decrease*^{1}
Derivationally Related Form	Lemmas with same morphological root	*destroy*^{1} ⟺ *destruction*^{1}

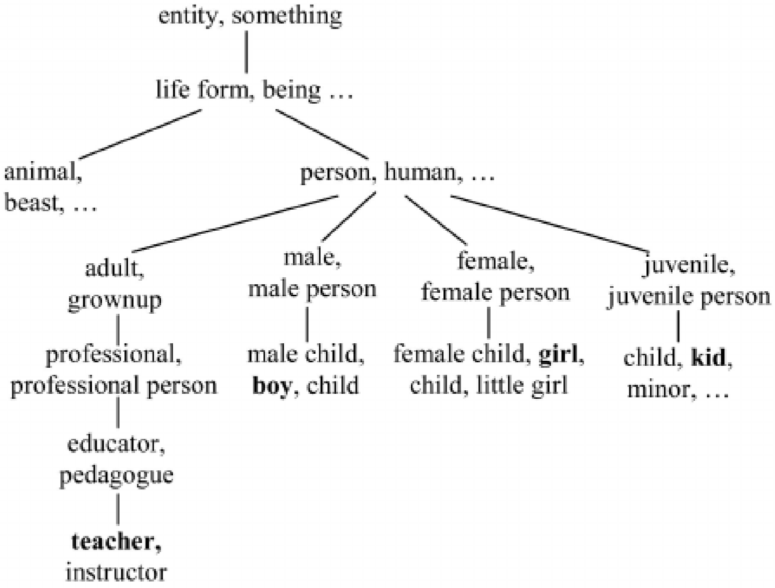
Other WordNet Hierarchy Fragment Examples



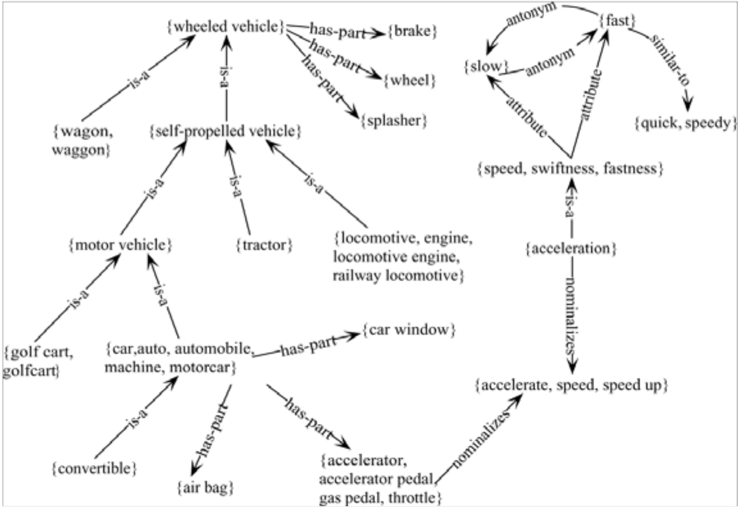
Other WordNet Hierarchy Fragment Examples



Other WordNet Hierarchy Fragment Examples



WordNet as as Graph



Supersenses in WordNet

Super senses are top-level hypernyms in the hierarchy.

Noun		
GROUP	1469	*place*
PERSON	1202	*people*
ARTIFACT	971	*car*
COGNITION	771	*way*
FOOD	766	*food*
ACT	700	*service*
LOCATION	638	*area*
TIME	530	*day*
EVENT	431	*experience*
COMMUNIC.*	417	*review*
POSSESSION	339	*price*
ATTRIBUTE	205	*quality*
QUANTITY	102	*amount*
ANIMAL	88	*dog*
BODY	87	*hair*
STATE	56	*pain*
NATURAL OBJ.	54	*flower*
RELATION	35	*portion*
SUBSTANCE	34	*oil*
FEELING	34	*discomfort*
PROCESS	28	*process*
MOTIVE	25	*reason*
PHENOMENON	23	*result*
SHAPE	6	*square*
PLANT	5	*tree*
OTHER	2	*stuff*

Verb		
STATIVE	2922	*is*
COGNITION	1093	*know*
COMMUNIC.*	974	*recommend*
SOCIAL	944	*use*
MOTION	602	*go*
POSSESSION	309	*pay*
CHANGE	274	*fix*
EMOTION	249	*love*
PERCEPTION	143	*see*
CONSUMPTION	93	*have*
BODY	82	*get...done*
CREATION	64	*cook*
CONTACT	46	*put*
COMPETITION	11	*win*
WEATHER	0	—

WordNets for Other Languages

- `globalwordnet.org/wordnets-in-the-world/` lists WordNets for tens of languages.
- Many of these WordNets are linked through **ILI – Interlingual Index** numbers.

Word Similarity

- **Synonymy**: a binary relation
  - Two words are either synonymous or not
- **Similarity** (or **distance**): a looser metric
  - Two words are more similar if they share more features of meaning
- Similarity is properly a relation between **senses**
  - The word “bank” is not similar to the word “slope”
  - bank_1 is similar to fund_3
  - bank_2 is similar to slope_5
- But we will compute similarity over both words and senses.

Why Word Similarity?

- A practical component in lots of NLP tasks
  - Question answering
  - Natural language generation
  - Automatic essay grading
  - Plagiarism detection
- A theoretical component in many linguistic and cognitive tasks
  - Historical semantics
  - Models of human word learning
  - Morphology and grammar induction

Similarity and Relatedness

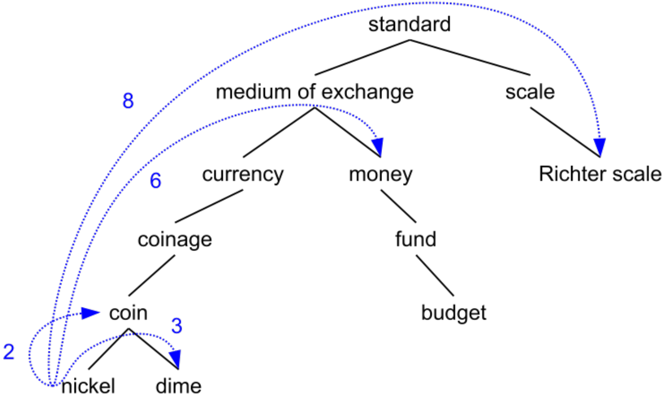
- We often distinguish **word similarity** from **word relatedness**
- **Similar words**: near-synonyms
- **Related words**: can be related in any way
- *car*, *bicycle*: **similar**
- *car*, *gasoline*: **related**, not similar

Two Classes of Similarity Algorithms

- WordNet/Thesaurus-based algorithms
  - Are words “nearby” in hypernym hierarchy?
  - Do words have similar glosses (definitions)?
- Distributional algorithms
  - Do words have similar distributional contexts?
  - Distributional (Vector) semantics.

Path-based Similarity

- Two concepts (senses/synsets) are similar if they are near each other in the hierarchy
  - They have a short path between them
  - Synsets have path 1 to themselves.



Refinements

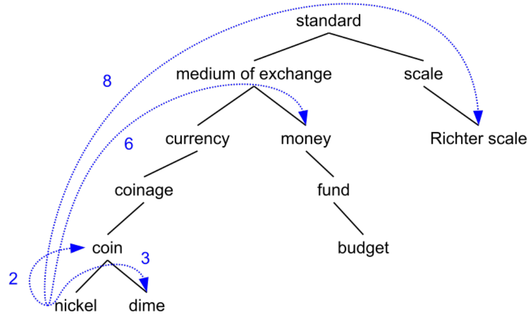
- $pathlen(c_1, c_2) = 1+$number of edges in the shortest path in the hypernym graph between sense nodes c_1 and c_2

- $simpath(c_1, c_2) = \dfrac{1}{pathlen(c_1, c_2)}$ (Ranges between 0 and 1)

- $wordsim(w_1, w_2) = \max\limits_{c_1 \in \text{senses}(w_1), c_2 \in \text{senses}(w_2)} simpath(c_1, c_2)$

Example for Path-based Similarity

- $simpath(nickel, coin) = 1/2 = .5$
- $simpath(fund, budget) = 1/2 = .5$
- $simpath(nickel, currency) = 1/4 = .25$
- $simpath(nickel, money) = 1/6 = .17$
- $simpath(coinage, Richterscale) = 1/6 = .17$



Problem with Basic Path-based Similarity

- Assumes each link represents a uniform distance
  - But *nickel* to *money* seems to us to be closer than *nickel* to *standard*
  - Nodes high in the hierarchy are very abstract
- We instead want a metric that
  - Represents the cost of each edge independently
  - Ranks words connected only through abstract nodes as less similar.

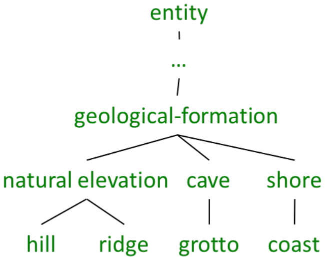
Information Content Similarity Metrics

- Define $p(c)$ as the probability that a randomly selected word in a corpus is an instance of concept c.
- Formally: there is a distinct random variable, ranging over words, associated with each concept in the hierarchy. For a given concept, each observed word/lemma is either
  - a member of that concept with probability $p(c)$
  - not a member of that concept with probability $1 - p(c)$
- All words are members of the root node (Entity): $p(root) = 1$
- The lower a node in hierarchy, the lower its probability

Information Content Similarity Metrics

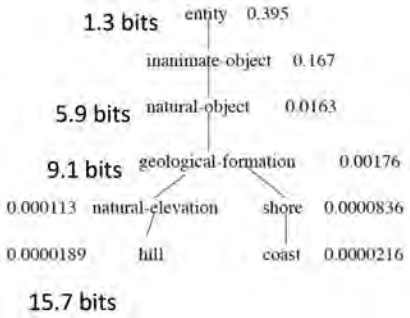
- Train by counting in a corpus
  - Each instance of *hill* counts toward frequency of *natural elevation*, *geological-formation*, *entity*, etc.
- Let $words(c)$ be the set of all words that are descendants of node c
  - $words($*geological-formation*$) = \{$*hill*, *ridge*, *grotto*, *coast*, *cave*, *shore*, *natural elevation*$\}$
  - $words($*natural elevation*$) = \{$*hill*, *ridge*$\}$
- For n words in the corpus

$$p(c) = \frac{\sum_{w \in words(c)} count(w)}{N}$$



Information Content: Definitions

- Information Content: $IC(c) = -\log p(c)$
- Most informative subsumer (lowest common subsumer)
  - $LCS(c_1, c_2) =$ The most informative (lowest) node in the hierarchy subsuming both c_1 and c_2.



The Resnik Method

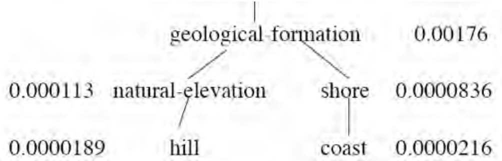
- The similarity between two words is related to their common information
- The more two words have in common, the more similar they are
- Resnik: measure common information as:

$$sim_{resnik}(c_1, c_2) = -\log p(LCS(c_1, c_2))$$

- $sim_{resnik}(hill, coast) = -\log p(LCS(hill, coast)) = -\log p(geological\text{-}formation) = 6.34$

The Dekang Lin Method

- The similarity between A and B is measured by the ratio between the amount of information needed to state the commonality of A and B and the information needed to fully describe what A and B are.
- $$sim_{lin}(c_1, c_2) = \frac{2 \log p(LCS(c_1, c_2))}{\log p(c_1) + \log p(c_2)}$$



- $sim_{lin}(hill, coast) = \frac{2 \log p(geological\text{-}formation)}{\log p(hill) + \log p(coast)} = 0.59$

Evaluating Similarity

- Extrinsic evaluation (Task-based)
  - Question answering
  - Essay grading
- Intrinsic evaluation: Correlation between algorithm and human word similarity ratings
  - Wordsim353 task: 353 noun pairs rated 0-10. $sim(\texttt{plane},\texttt{car}) = 5.77$
  - Taking TOEFL multiple-choice vocabulary tests
    - Levied is closest in meaning to: imposed, believed, requested, correlated.

11-411
Natural Language Processing
Distributional/Vector Semantics

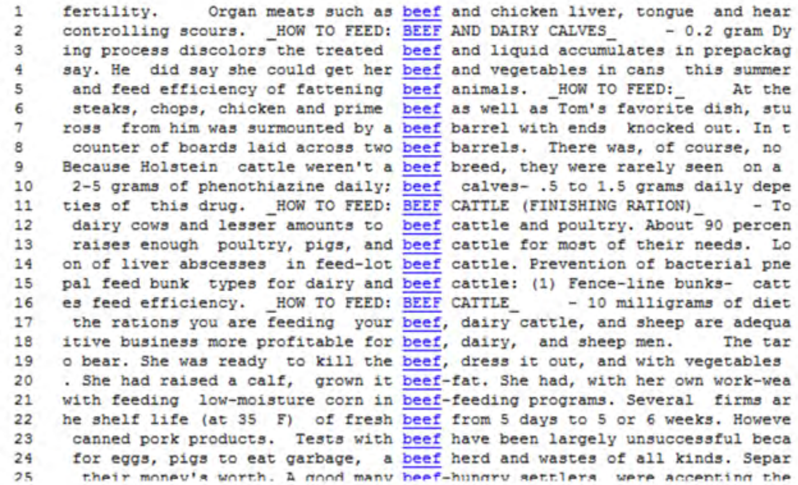
Kemal Oflazer

Carnegie Mellon University in Qatar

The Distributional Hypothesis

- Want to know the meaning of a word? Find what words occur with it.
  - Leonard Bloomfield
  - Edward Sapir
  - Zellig Harris–first formalization
    - “oculist and eye-doctor . . . occur in almost the same environments”
    - “If A and B have almost identical environments we say that they are synonyms.”
- The best known formulation comes from J.R. Firth:
  - “You shall know a word by the company it keeps.”

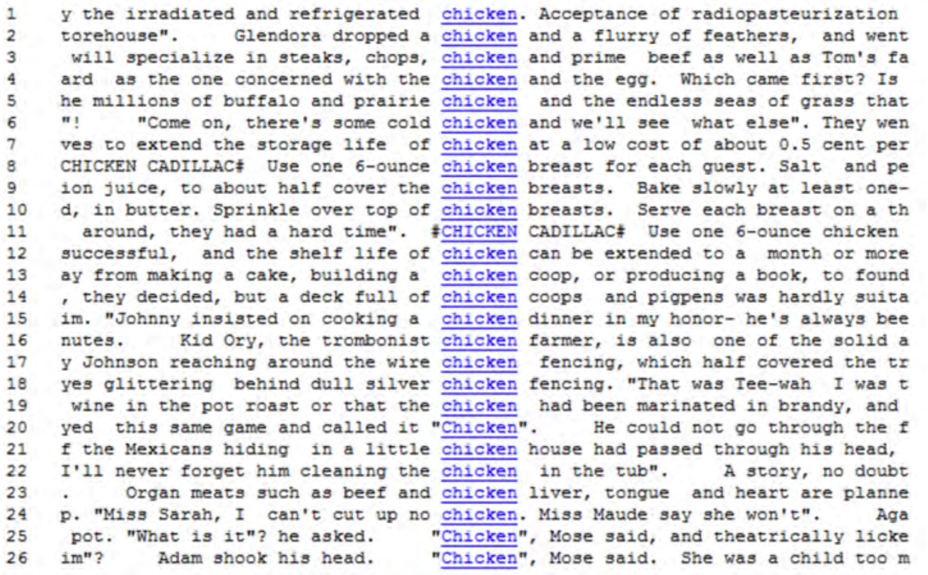
Contexts for *Beef*



```
1    fertility.     Organ meats such as beef and chicken liver, tongue  and hear
2    controlling scours.  _HOW TO FEED: BEEF AND DAIRY CALVES_     - 0.2 gram Dy
3    ing process discolors the treated  beef and liquid accumulates in prepackag
4    say. He  did say she could get her beef and vegetables in cans  this summer
5     and feed efficiency of fattening  beef animals.  _HOW TO FEED:_     At the
6     steaks, chops, chicken and prime  beef as well as Tom's favorite dish, stu
7    ross  from him was surmounted by a beef barrel with ends  knocked out. In t
8     counter of boards laid across two beef barrels.  There was, of course, no
9    Because Holstein  cattle weren't a beef breed, they were rarely seen  on a
10    2-5 grams of phenothiazine daily; beef  calves- .5 to 1.5 grams daily depe
11   ties of  this drug.  _HOW TO FEED: BEEF CATTLE (FINISHING RATION)_    - To
12    dairy cows and lesser amounts to  beef cattle and poultry. About 90 percen
13    raises enough  poultry, pigs, and beef cattle for most of their needs.  Lo
14   on of liver abscesses  in feed-lot beef cattle. Prevention of bacterial pne
15   pal feed bunk  types for dairy and beef cattle: (1) Fence-line bunks-  catt
16   es feed efficiency.  _HOW TO FEED: BEEF CATTLE_     - 10 milligrams of diet
17    the rations you are feeding  your beef, dairy cattle, and sheep are adequa
18   itive business more profitable for beef, dairy,  and sheep men.     The tar
19   o bear. She was ready  to kill the beef, dress it out, and with vegetables
20   . She had raised a calf,  grown it beef-fat. She had, with her own work-wea
21   with feeding  low-moisture corn in beef-feeding programs. Several  firms ar
22   he shelf life (at 35  F)  of fresh beef from 5 days to 5 or 6 weeks. Howeve
23    canned pork products.  Tests with beef have been largely unsuccessful beca
24    for eggs, pigs to eat garbage,  a beef herd and wastes of all kinds. Separ
25     their money's worth. A good many beef-hungry settlers  were accepting the
```

- This is called a *concordance*.

Contexts for *Chicken*



```
1    y the irradiated and refrigerated  chicken. Acceptance of radiopasteurization
2    torehouse".    Glendora dropped a chicken and a flurry of feathers,  and went
3     will specialize in steaks, chops, chicken and prime  beef as well as Tom's fa
4    ard  as the one concerned with the chicken and the egg.  Which came first? Is
5    he millions of buffalo and prairie chicken  and the endless seas of grass that
6    "!     "Come on, there's some cold chicken and we'll see  what else". They wen
7    ves to extend the storage life  of chicken at a low cost of about 0.5 cent per
8    CHICKEN CADILLAC#  Use one 6-ounce chicken breast for each guest. Salt  and pe
9    ion juice, to about half cover the chicken breasts.  Bake slowly at least one-
10   d, in butter. Sprinkle over top of chicken breasts.  Serve each breast on a th
11     around, they had a hard time".  #CHICKEN CADILLAC#  Use one 6-ounce chicken
12   successful,  and the shelf life of chicken can be extended to a  month or more
13   ay from making a cake, building a  chicken coop, or producing a book, to found
14   , they decided, but a deck full of chicken coops  and pigpens was hardly suita
15   im. "Johnny insisted on cooking a  chicken dinner in my honor- he's always bee
16   nutes.    Kid Ory, the trombonist chicken farmer, is also  one of the solid a
17   y Johnson reaching around the wire chicken  fencing, which half covered the tr
18   yes glittering  behind dull silver chicken fencing. "That was Tee-wah  I was t
19    wine in the pot roast or that the chicken  had been marinated in brandy, and
20   yed  this same game and called it "Chicken".     He could not go through the f
21   f the Mexicans hiding  in a little chicken house had passed through his head,
22   I'll never forget him cleaning the chicken  in the tub".     A story, no doubt
23   .     Organ meats such as beef and chicken liver, tongue  and heart are planne
24   p. "Miss Sarah, I  can't cut up no chicken. Miss Maude say she won't".     Aga
25    pot. "What is it"? he asked.     "Chicken", Mose said, and theatrically licke
26   im"?    Adam shook his head.     "Chicken", Mose said.  She was a child too m
```

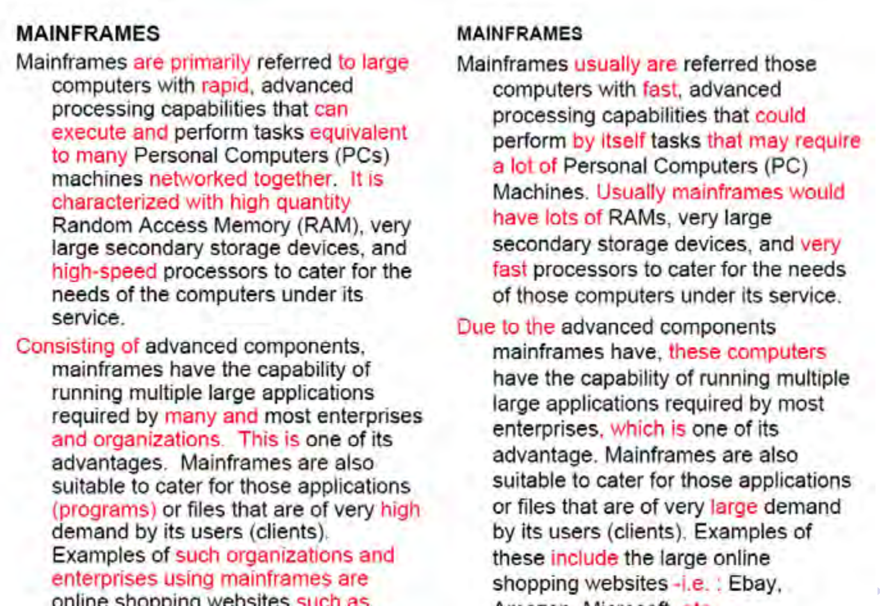
Intuition for Distributional Word Similarity

- Consider
  - A bottle of pocarisweat is on the table.
  - Everybody likes pocarisweat.
  - Pocarisweat makes you feel refreshed.
  - They make pocarisweat out of ginger.
- From context words humans can guess **pocarisweat** means a beverage like **coke**.
- So the intuition is that two words are similar if they have similar word contexts.

Why Vector Models of Meaning?

- Computing similarity between words:
  - *fast* is similar to *rapid*
  - *tall* is similar to *height*
- Application: Question answering:
  - Question: “How *tall* is Mt. Everest?”
  - Candidate A: “The official *height* of Mount Everest is 29029 feet.”

Word Similarity for Plagiarism Detection



MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high demand by its users (clients). Examples of such organizations and enterprises using mainframes are online shopping websites such as

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand by its users (clients). Examples of these include the large online shopping websites -i.e. : Ebay, Amazon, Microsoft, etc

Vector Models

- Sparse vector representations:
  - Mutual-information weighted word co-occurrence matrices.
- Dense vector representations:
  - Singular value decomposition (and Latent Semantic Analysis)
  - Neural-network-inspired models (skip-grams, CBOW)
  - Brown clusters

Shared Intuition

- Model the meaning of a word by “embedding” in a vector space.
- The meaning of a word is a vector of numbers:
  - Vector models are also called **embeddings**.
- In contrast, word meaning is represented in many (early) NLP applications by a vocabulary index (“word number 545”)

Term-document Matrix

- Each cell is the count of term *t* in a document *d* ($tf_{t,d}$).
- Each document is a count vector in $\mathbb{N}^V$, a column below.

	As You Like It	**Twelfth Night**	**Julius Caesar**	**Henry V**
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document Matrix

- Two documents are similar of their vectors are similar.

	As You Like It	**Twelfth Night**	**Julius Caesar**	**Henry V**
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document Matrix

- Each word is a count vector in $\mathbb{N}^D$ – a row below

	As You Like It	**Twelfth Night**	**Julius Caesar**	**Henry V**
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document Matrix

- Two words are similar if their vectors are similar.

	As You Like It	**Twelfth Night**	**Julius Caesar**	**Henry V**
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-context Matrix for Word Similarity

- Two words are similar if their **context vectors** are similar.

	aardvark	**computer**	**data**	**pinch**	**result**	**sugar . . .**
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

Word–Word or Word–Context Matrix

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of ± 4 words
- A word is now defined by a vector over counts of words in context.
  - If a word w_j occurs in the context of w_i, increase $count_{ij}$.
- Assuming we have V words,
  - Each vector is now of length V.
  - The word-word matrix is $V \times V$.

Sample Contexts of ±7 Words

sugar, a sliced lemon, a tablespoonful of	**apricot**	preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

	aardvark	**computer**	**data**	**pinch**	**result**	**sugar . . .**
⋮						
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0
⋮						

The Word–Word Matrix

- We showed only a 4×6 matrix, but the real matrix is $50,000 \times 50,000$.
  - So it is very sparse: Most values are 0.
  - That's OK, since there are lots of efficient algorithms for sparse matrices.
- The size of windows depends on the goals:
  - The smaller the context $(\pm 1 - 3)$, the more syntactic the representation
  - The larger the context $(\pm 4 - 10)$, the more syntactic the representation

Types of Co-occurence between Two Words

- First-order co-occurrence (syntagmatic association):
  - They are typically nearby each other.
  - *wrote* is a first-order associate of *book* or *poem*.
- Second-order co-occurrence (paradigmatic association):
  - They have similar neighbors.
  - *wrote* is a second-order associate of words like *said* or *remarked*.

Problem with Raw Counts

- Raw word frequency is not a great measure of association between words.
  - It is very skewed: “the” and “of” are very frequent, but maybe not the most discriminative.
- We would rather have a measure that asks whether a context word is **particularly informative** about the target word.
  - Positive Pointwise Mutual Information (PPMI)

Pointwise Mutual Information

- **Pointwise Mutual Information**: Do events x and y co-occur more that if they were independent.

$$\text{PMI}(x,y) = \log_2 \frac{p(x,y)}{p(x)p(y)}$$

- **PMI between two words**: Do target word w and context word c co-occur more that if they were independent.

$$\text{PMI}(w,c) = \log_2 \frac{p(w,c)}{p(w)p(c)}$$

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic:
  - Things are co-occurring less than we expect by chance
  - Unreliable without enormous corpora
    - Imagine $w-1$ and w_2 whose probability is each 10^{-6}.
    - Hard to be sure $p(w_1, w_2)$ is significantly different than 10^{-12}.
  - Furthermore it’s not clear people are good at “unrelatedness”.
- So we just replace negative PMI values by 0.

$$\text{PPMI}(w,c) = max\left(\log_2 \frac{p(w,c)}{p(w)p(c)}, 0\right)$$

Computing PPMI on a Term-Context Matrix

- We have matrix F with V rows (words) and C columns (contexts) (in general $C = V$)
- f_{ij} is how many times word w_i co-occurs in the context of the word c_j.

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{V}(\sum_{j=1}^{C} f_{ij})}$$

$$p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{V}(\sum_{j=1}^{C} f_{ij})} \qquad p_{*j} = \frac{\sum_{i=1}^{V} f_{ij}}{\sum_{i=1}^{V}(\sum_{j=1}^{C} f_{ij})}$$

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \text{max}(pmi_{ij}, 0)$$

Example

	computer	**data**	**pinch**	**result**	**sugar**	
apricot	0	0	1	0	1	2
pineapple	0	0	1	0	1	2
digital	2	1	0	1	0	4
information	1	6	0	4	0	11
	3	7	2	5	2	19

$$p(w = information, c = data) = \frac{6}{19} = 0.32$$

$$p(w = information) = \frac{11}{19} = 0.58 \quad p(c = data) = \frac{7}{19} = 0.32$$

$p(w, c)$

	computer	**data**	**pinch**	**result**	**sugar**	$p(w)$
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
$p(c)$	0.16	0.37	0.11	0.26	0.11	

Example

$p(w,c)$

	computer	**data**	**pinch**	**result**	**sugar**	$p(w)$
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
$p(c)$	0.16	0.37	0.11	0.26	0.11	

$$pmi(information, data) = \log_2 \frac{0.32}{0.37 \cdot 0.57} \approx 0.58$$

$PPMI(w,c)$

	computer	**data**	**pinch**	**result**	**sugar**
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.32	-	0.47	-

Issues with PPMI

- PMI is biased toward infrequent events.
- Very rare words have very high PMI values.
- Two solutions:
  - Give rare words slightly higher probabilities
  - Use add-one smoothing (which has a similar effect)

Issues with PPMI

- Raise the context probabilities to $\alpha = 0.75$:

$$\text{PPMI}_{\alpha}(w, c) = \max(\log_2 \frac{p(w, c)}{p(w)p_{\alpha}(c)}, 0)$$

$$p_{\alpha}(c) = \frac{count(c)^{\alpha}}{\sum_c count(c)^{\alpha}}$$

- This helps because $p_{\alpha}(c) > p(c)$ for rare c.
- Consider two context words $p(a) = 0.99$ and $p(b) = 0.01$
- $p_{\alpha}(a) = \frac{0.99^{0.75}}{0.99^{0.75}+0.01^{0.75}} = 0.97 \qquad p_{\alpha}(b) = \frac{0.99^{0.75}}{0.99^{0.75}+0.01^{0.75}} = 0.03$

Using Laplace Smoothing

Add-2 Smoothed $Count(w, c)$

	computer	**data**	**pinch**	**result**	**sugar**
apricot	2	2	3	2	3
pineapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

$p(w, c)$ Add-2

	computer	**data**	**pinch**	**result**	**sugar**	$p(w)$
apricot	0.03	0.03	0.05	0.03	0.05	0.20
pineapple	0.03	0.03	0.05	0.03	0.05	0.20
digital	0.07	0.05	0.03	0.05	0.03	0.24
information	0.05	0.14	0.03	0.10	0.03	0.36
$p(c)$	0.19	0.25	0.17	0.22	0.17	

PPMI vs. add-2 Smoothed PPMI

$PPMI(w,c)$

	computer	**data**	**pinch**	**result**	**sugar**	
apricot	-	-	2.25	-	2.25	
pineapple	-	-	2.25	-	2.25	
digital	1.66	0.00	-	0.00	-	
information	0.00	0.32	-	0.47	-	-

$PPMI(w,c)$

	computer	**data**	**pinch**	**result**	**sugar**
apricot	0.00	0.00	0.56	0.00	0.56
pineapple	0.00	0.00	0.56	0.00	0.56
digital	0.62	0.00	0.00	0.00	0.00
information	0.00	0.58	0.00	0.37	0.00

Measuring Similarity

- Given two target words represented with vectors $\boldsymbol{v}$ and $\boldsymbol{w}$.
- The **dot product** or **inner product** is usually used basis for similarity.

$$\boldsymbol{v} \cdot \boldsymbol{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \cdots + v_N w_N = |\boldsymbol{v}||\boldsymbol{w}| \cos\theta$$

- $\boldsymbol{v} \cdot \boldsymbol{w}$ is high when two vectors have large values in the same dimensions.
- $\boldsymbol{v} \cdot \boldsymbol{w}$ is low (in fact 0) with zeros in complementary distribution.
- We also do not want the similarity to be sensitive to word-frequency.
- So normalize by vector length and use the cosine as the similarity
  $\cos(\boldsymbol{v}, \boldsymbol{w}) = \frac{\boldsymbol{v} \cdot \boldsymbol{w}}{|\boldsymbol{v}||\boldsymbol{w}|}$

Other Similarity Measures in the Literature

- $$\text{sim}_{cosine}(\boldsymbol{v}, \boldsymbol{w}) = \frac{\boldsymbol{v} \cdot \boldsymbol{w}}{|\boldsymbol{v}||\boldsymbol{w}|}$$

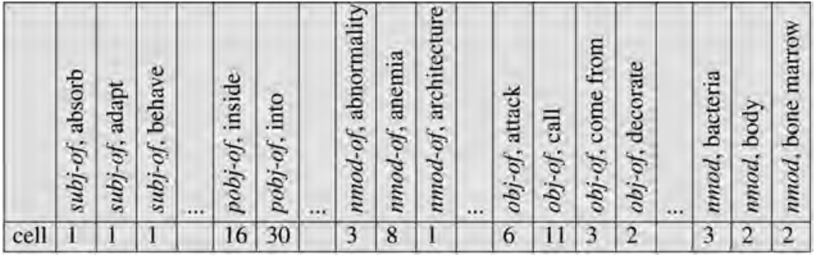
- $$\text{sim}_{Jaccard}(\boldsymbol{v}, \boldsymbol{w}) = \frac{\sum_i \min(v_i, w_i)}{\sum_i \max(v_i, w_i)}$$

- $$\text{sim}_{Dice}(\boldsymbol{v}, \boldsymbol{w}) = \frac{2 \sum_i \min(v_i, w_i)}{\sum_i (v_i + w_i)}$$

Using Syntax to Define Context

- “The meaning of entities, and the meaning of grammatical relations among them, is related to the restriction of combinations of these entities relative to other entities.” (Zelig Harris (1968))
- Two words are similar if they appear in similar syntactic contexts.
  - *duty* and *responsibility* have similar syntactic distribution
    - **Modified by Adjectives**: *additional*, *administrative*, *assumed*, *collective*, *congressional*, *constitutional*, . . .
    - **Objects of Verbs**: *assert*, *assign*, *assume*, *attend to*, *avoid*, *become*, *breach*, . . .

Co-occurence Vectors based on Syntactic Dependencies



	subj-of, absorb	*subj-of*, adapt	*subj-of*, behave	...	*pobj-of*, inside	*pobj-of*, into	...	*nmod-of*, abnormality	*nmod-of*, anemia	*nmod-of*, architecture	...	*obj-of*, attack	*obj-of*, call	*obj-of*, come from	*obj-of*, decorate	...	*nmod*, bacteria	*nmod*, body	*nmod*, bone marrow
cell	1	1	1		16	30		3	8	1		6	11	3	2		3	2	2

- Each context dimension is a context word in one of R grammatical relations.
  - Each word vector now has $R \cdot V$ entries.
- Variants have V dimensions with the count being total for R relations.

Sparse vs. Dense Vectors

- PPMI vectors are
  - long (length in 10s of thousands)
  - sparse (most elements are 0)
- Alternative: learn vectors which are
  - short (length in several hundreds)
  - dense (most elements are non-zero)

Why Dense Vectors?

- Short vectors may be easier to use as features in machine learning (less weights to tune).
- Dense vectors may generalize better than storing explicit counts.
- They may do better at capturing synonymy:
  - *car* and *automobile* are synonyms
  - But they are represented as distinct dimensions
  - This fails to capture similarity between a word with *car* as a neighbor and a word with *automobile* as a neighbor

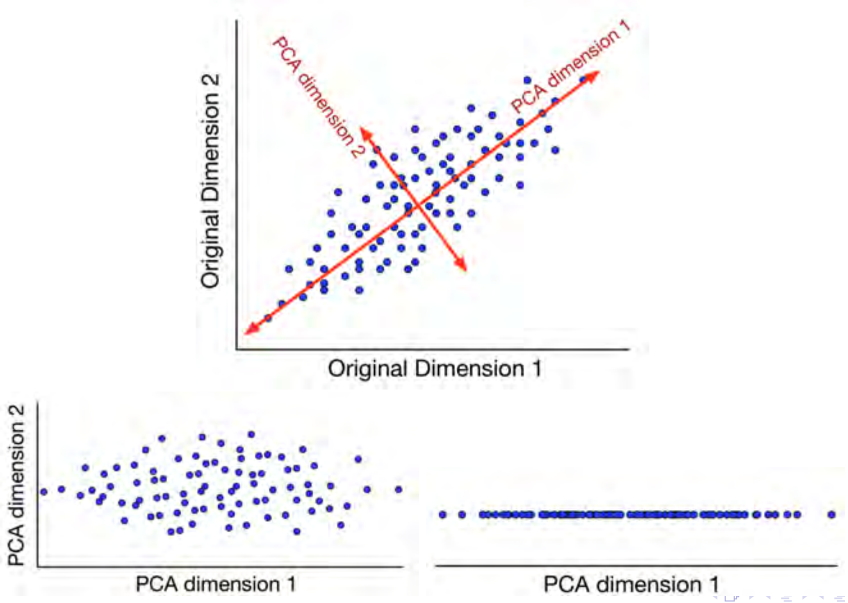
Methods for Getting Short Dense Vectors

- Singular Value Decomposition (SVD)
  - A special case of this is called LSA - Latent Semantic Analysis
- “Neural Language Model”-inspired predictive models.
  - skip-grams and continuous bag-of-words (CBOW)
- Brown clustering

Dense Vectors via SVD - Intuition

- Approximate an N-dimensional dataset using fewer dimensions
- By rotating the axes into a new space along the dimension with the most variance
- Then repeat with the next dimension captures the next most variance, etc.
- Many such (related) methods:
  - PCA - principle components analysis
  - Factor Analysis
  - SVD

Dimensionality Reduction



Singular Value Decomposition

- Any square $v \times v$ matrix (of rank v) X equals the product of three matrices.

$$\underset{v \times v}{\overset{X}{\begin{bmatrix} x_{11} & \dots & x_{1v} \\ x_{21} & \dots & x_{2v} \\ \vdots & \ddots & \vdots \\ x_{v1} & \dots & x_{vv} \end{bmatrix}}} = \underset{v \times v}{\overset{W}{\begin{bmatrix} w_{11} & \dots & w_{1m} \\ \vdots & \ddots & \vdots \\ w_{v1} & \dots & w_{vv} \end{bmatrix}}} \times \underset{v \times v}{\overset{S}{\begin{bmatrix} \sigma_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_{vv} \end{bmatrix}}} \times \underset{v \times v}{\overset{C}{\begin{bmatrix} c_{11} & \dots & c_{1c} \\ \vdots & \ddots & \vdots \\ c_{m1} & \dots & c_{vv} \end{bmatrix}}}$$

- v columns in W are orthogonal to each other and are ordered by the amount of variance each new dimension accounts for.
- S is a diagonal matrix of eigenvalues expressing the importance of each dimension.
- C has v rows for the singular values and v columns corresponding to the original contexts.

Reducing Dimensionality with Truncated SVD

$$\underset{v \times v}{\overset{X}{\begin{bmatrix} x_{11} & \dots & x_{1c} \\ x_{21} & \dots & x_{2c} \\ \vdots & \ddots & \vdots \\ x_{v1} & \dots & x_{vv} \end{bmatrix}}} = \underset{v \times v}{\overset{W}{\begin{bmatrix} w_{11} & \dots & w_{1m} \\ \vdots & \ddots & \vdots \\ w_{v1} & \dots & w_{vv} \end{bmatrix}}} \times \underset{v \times v}{\overset{S}{\begin{bmatrix} \sigma_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_{vv} \end{bmatrix}}} \times \underset{v \times v}{\overset{C}{\begin{bmatrix} c_{11} & \dots & c_{1c} \\ \vdots & \ddots & \vdots \\ c_{m1} & \dots & c_{vv} \end{bmatrix}}}$$

$$\Downarrow$$

$$\underset{v \times c}{\overset{X}{\begin{bmatrix} x_{11} & \dots & x_{1v} \\ x_{21} & \dots & x_{2v} \\ \vdots & \ddots & \vdots \\ x_{v1} & \dots & x_{vv} \end{bmatrix}}} \approx \underset{v \times k}{\overset{W}{\begin{bmatrix} w_{11} & \dots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{v1} & \dots & w_{vk} \end{bmatrix}}} \times \underset{k \times k}{\overset{S}{\begin{bmatrix} \sigma_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_{kk} \end{bmatrix}}} \times \underset{k \times v}{\overset{C}{\begin{bmatrix} c_{11} & \dots & c_{1v} \\ \vdots & \ddots & \vdots \\ c_{m1} & \dots & c_{kv} \end{bmatrix}}}$$

Truncated SVD Produces Embeddings

$$\begin{bmatrix} w_{11} & \cdots & w_{1k} \\ w_{21} & \cdots & w_{2k} \\ \vdots & \ddots & \vdots \\ w_{v1} & \cdots & w_{vk} \end{bmatrix}$$

- Each row of W matrix is a k-dimensional representation of each word w.
- k may range from 50 to 1000

Embeddings vs Sparse Vectors

- Dense SVD embeddings sometimes work better than sparse PPMI matrices at tasks like word similarity
- Denoising: low-order dimensions may represent unimportant information
- Truncation may help the models generalize better to unseen data.
- Having a smaller number of dimensions may make it easier for classifiers to properly weight the dimensions for the task.
- Dense models may do better at capturing higher order co-occurrence.

Embeddings Inspired by Neural Language Models

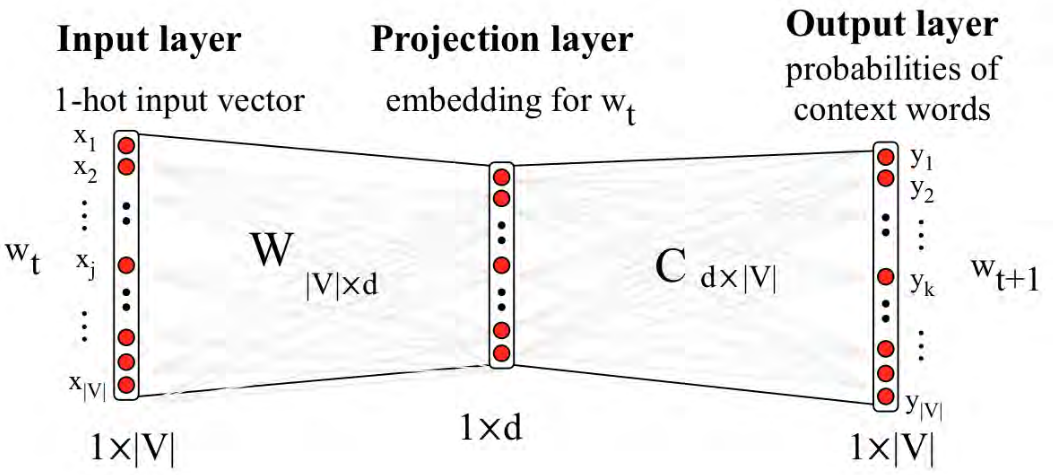
- Skip-gram and CBOW learn embeddings as part of the process of word prediction.
- Train a neural network to predict neighboring words
  - Inspired by neural net language models.
  - In so doing, learn dense embeddings for the words in the training corpus.
- Advantages:
  - Fast, easy to train (much faster than SVD).
  - Available online in the `word2vec` package.
  - Including sets of pretrained embeddings!

Skip-grams

- From the current word w_t, predict other words in a context window of $2C$ words.
- For example, we are given w_t and we are predicting one of the words in

$$[w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}]$$

Compressing Words

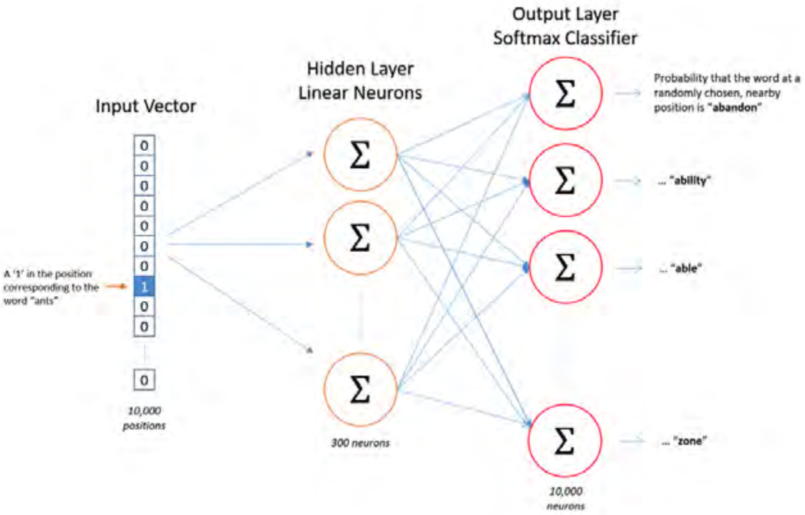


One-hot Vector Representation

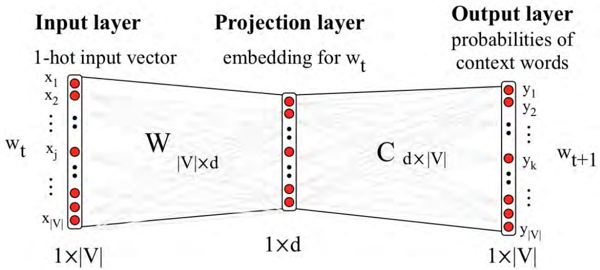
- Each word in the vocabulary is represented with a vector of length $|V|$.
  - 1 for the index target word w_t and 0 for other words.
- So if “popsicle” is vocabulary word 5, the one-hot vector is

$$[0, 0, 0, 0, 1, 0, 0, 0, 0, \ldots, 0]$$

Neural Network Architecture



Where are the Word Embeddings?



- The rows of the first matrix actually are the word embeddings.
- Multiplication of the one-hot input vector “selects” the relevant row as the output to hidden layer.

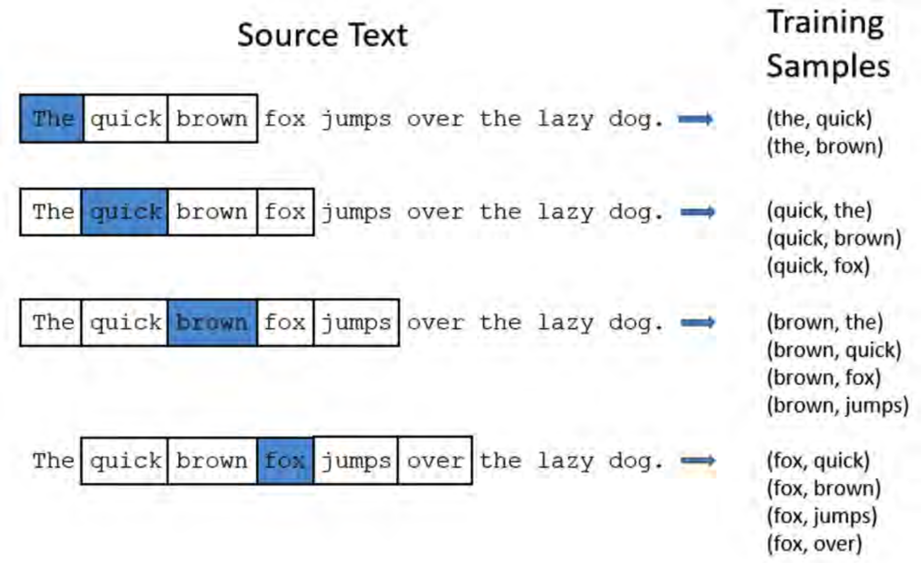
$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = \begin{bmatrix} 10 & 12 & 19 \end{bmatrix}$$

Output Probabilities

- The output vector is also a vector (hidden-layer) and matrix multiplication (the C matrix).
  - The value computed for output unit $k = c_k \cdot w_j$ where w_j is the hidden layer vector (for word j).
- Except, the outputs are not probabilities!
- We use the same scaling idea we used earlier and then use *softmax* .

$$p(w_k \text{ is in the context of } w_j) = \frac{exp(c_k \cdot v_j)}{\sum_i exp(c_i \cdot v_j)}$$

Training for Embeddings



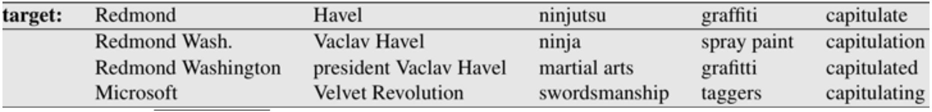
Source Text	Training Samples
The quick brown fox jumps over the lazy dog. →	(the, quick) (the, brown)
The quick brown fox jumps over the lazy dog. →	(quick, the) (quick, brown) (quick, fox)
The quick brown fox jumps over the lazy dog. →	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown fox jumps over the lazy dog. →	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

Training for Embeddings

- You have a huge network (say you have 1M words and embedding dimension of 300).
  - You have 300M entries in each of the matrices.
- Running gradient descent (via *backpropagation*) is very slow.
- Some innovations used:
  - Reduce vocabulary for phrases like “New York”
  - Reduce vocabulary and training samples by ignoring infrequent words.
  - Instead of computing probabilities through the expensive scaling process, use *negative sampling* to only update a small number of the weights each time.
- It turns out these improve the quallity of the vectors also!

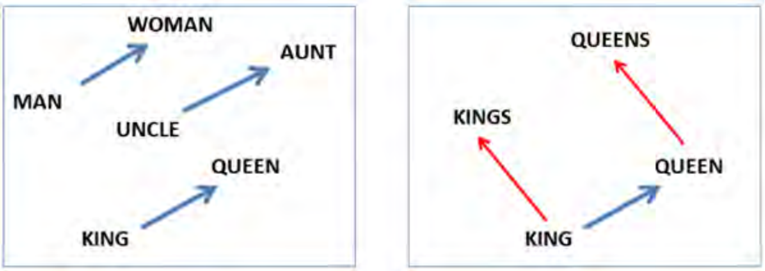
Properties of Embeddings

- Nearest words to some embeddings in the $d-$ dimensional space.



target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

- Relation meanings
  - $vector(king) - vector(man) + vector(woman) \approx vector(queen)$
  - $vector(Paris) - vector(France) + vector(Italy) \approx vector(Rome)$



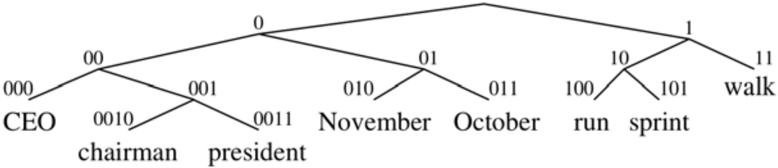
Brown Clustering

- An agglomerative clustering algorithm that clusters words based on which words precede or follow them
- These word clusters can be turned into a kind of vector
- We will do a brief outline here.

Brown Clustering

- Each word is initially assigned to its own cluster.
- We now consider consider merging each pair of clusters. Highest quality merge is chosen.
  - Quality = merges two words that have similar probabilities of preceding and following words.
  - More technically quality = smallest decrease in the likelihood of the corpus according to a class-based language model
- Clustering proceeds until all words are in one big cluster.

Brown Clusters as Vectors



- By tracing the order in which clusters are merged, the model builds a binary tree from bottom to top.
- Each word represented by binary string = path from root to leaf
- Each intermediate node is a cluster
- Chairman represented by 0010, “months” by 01, and verbs by 1.

Class-based Language Model

- Suppose each word is in some class c_i.

$$p(w_i \mid w_{i-1}) = p(c_i \mid c_{i-1})\, p(w_i \mid c_i)$$

11-411
Natural Language Processing
Word Sense Disambiguation

Kemal Oflazer

Carnegie Mellon University in Qatar

Homonymy and Polysemy

- As we have seen, multiple words can be spelled the same way (*homonymy*; technically *homography*)
- The same word can also have different, related senses (*polysemy*)
- Various NLP tasks require resolving the ambiguities produced by homonymy and polysemy.
- Word sense disambiguation (WSD)

Versions of the WSD Task

- Lexical sample
  - Choose a sample of words.
  - Choose a sample of senses for those words.
  - Identify the right sense for each word in the sample.
- All-words
  - Systems are given the entire text.
  - Systems are given a lexicon with senses for every content word in the text.
  - Identify the right sense for each content word in the text .

Supervised WSD

- If we have hand-labelled data, we can do supervised WSD.
- Lexical sample tasks
  - Line-hard-serve corpus
  - SENSEVAL corpora
- All-word tasks
  - Semantic concordance: SemCor – subset of Brown Corpus manually tagged with WordNet senses.
  - SENSEVAL-3
- Can be viewed as a classification task

Sample SemCor Data

```
<wf cmd=done pos=PRP$ ot=notag>Your</wf>
<wf cmd=done pos=NN lemma=invitation wnsn=1 lexsn=1:10:00::>invitation</wf>
<wf cmd=ignore pos=TO>to</wf>
<wf cmd=done pos=VB lemma=write_about wnsn=1 lexsn=2:36:00::>write_about</wf>
<wf cmd=done rdf=person pos=NNP lemma=person wnsn=1 lexsn=1:03:00:: pn=person>S
<wf cmd=ignore pos=TO>to</wf>
<wf cmd=done pos=VB lemma=honor wnsn=1 lexsn=2:41:00::>honor</wf>
<wf cmd=ignore pos=PRP$>his</wf>
<wf cmd=done pos=JJ lemma=70th wnsn=1 lexsn=5:00:00:ordinal:00>70_th</wf>
<wf cmd=done pos=NN lemma=anniversary wnsn=1 lexsn=1:28:00::>Anniversary</wf>
<wf cmd=ignore pos=IN>for</wf>
<wf cmd=ignore pos=DT>the</wf>
<wf cmd=done pos=NN lemma=april wnsn=1 lexsn=1:28:00::>April</wf>
<wf cmd=done pos=NN lemma=issue wnsn=2 lexsn=1:10:00::>issue</wf>
<wf cmd=ignore pos=IN>of</wf>
<wf cmd=done pos=NNP pn=other ot=notag>Sovietskaya_Muzyka</wf>
<wf cmd=done pos=VBZ ot=notag>is</wf>
<wf cmd=done pos=VB lemma=accept wnsn=6 lexsn=2:40:01::>accepted</wf>
<wf cmd=ignore pos=IN>with</wf>
<wf cmd=done pos=NN lemma=pleasure wnsn=1 lexsn=1:12:00::>pleasure</wf>
```

What Features Should One Use?

- Warren Weaver commented in 1955

  If one examines the words in a book, one at a time as through an opaque mask with a hole in it one word wide, then it is obviously impossible to determine, one at a time, the meaning of the words. [. . .] But if one lengthens the slit in the opaque mask, until one can see not only the central word in question but also say N words on either side, then if N is large enough one can unambiguously decide the meaning of the central word. [. . .] The practical question is: "What minimum value of N will, at least in a tolerable fraction of cases, lead to the correct choice of meaning for the central word?"

- What information is available in that window of length N that allows us to do WSD?

What Features Should One Use?

- Collocation features
  - Encode information about specific positions located to the left or right of the target word
  - For example $[w_{i-2}, POS_{i-2}, w_{i-1}, POS_{i-2}, w_{i+1}, POS_{i+1}, w_{i+2}, POS_{i+2}]$
  - For *bass*, e.g., $[guitar, NN, and, CC, player, NN, stand, VB]$
- Bag-of-words features
  - Unordered set of words occurring in window
  - Relative sequence is ignored
  - Words are lemmatized
  - Stop/Function words typically ignored.
  - Used to capture domain

Naive Bayes for WSD

- Choose the most probable sense given the feature vector $\boldsymbol{f}$ which can be formulated into

$$\hat{s} = \arg\max_{s \in S} \; p(s) \prod_{j=1}^{n} p(f_j \mid s)$$

- Naive Bayes assumes features in $\boldsymbol{f}$ are independent (often not true)
- But usually Naive Bayes Classifiers perform well in practice.

Semisupervised WSD–Decision List Classifiers

- The decisions handed down by naive Bayes classifiers (and other similar ML algorithms) are difficult to interpret.
  - It is not always clear why, for example, a particular classification was made.
  - For reasons like this, some researchers have looked to decision list classifiers, a highly interpretable approach to WSD .
- We have a list of *conditional* statements.
  - Item being classified falls through the cascade until a statement is true.
  - The associated sense is then returned.
  - Otherwise, a default sense is returned.
- Where does the list come from?

Decision List Features for WSD – Collocational Features

- Word immediately to the left or right of target:
  - I have my bank$_1$ *statement*.
  - The *river* bank$_2$ is muddy.
- Pair of words to immediate left or right of target:
  - The *world's richest* bank$_1$ is here in New York.
  - The river bank$_2$ *is muddy*.
- Words found within *k* positions to left or right of target, where k is often 10-50 :
  - My *credit* is just horrible because my bank$_1$ has made several mistakes with my *account* and the *balance* is very low.

Learning a Decision List Classifier

- Each individual feature-value is a test.
- Features exists in a small context around the word.
- How discriminative is a feature among the senses?
- For a given ambiguous word compute

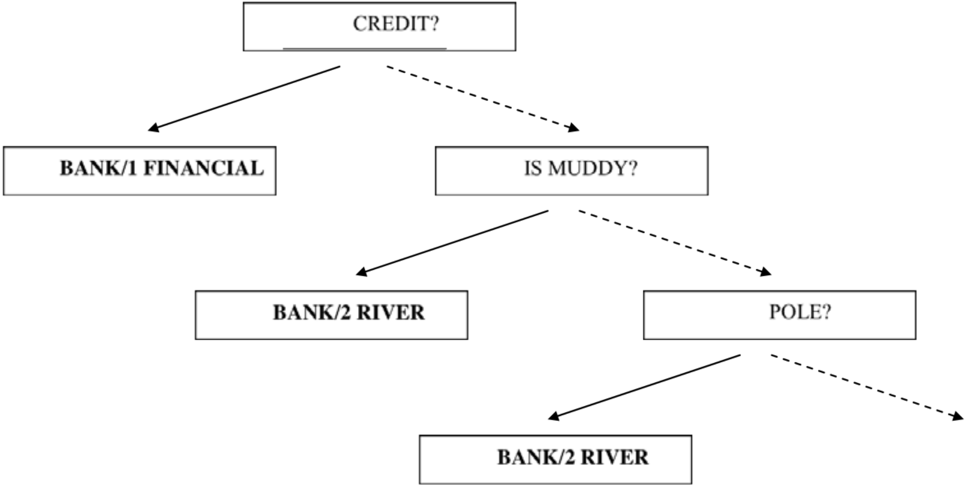
  $$weight(s_i, f_j) = \log \frac{p(s_i \mid f_j)}{p(\neg s_i \mid f_j)}$$

  where $\neg s_i$ all other senses of the word except s_i.
- Order in descending order ignoring values ≤ 0.
- When testing the first test that matches gives the sense.

Example

- Given 2,000 instances of “bank”, 1,500 for bank/1 (financial sense) and 500 for bank/2 (river sense)
  - $p(s_1) = 1,500/2,000 = .75$
  - $p(s_2) = 500/2,000 = .25$
- Given “credit” occurs 200 times with bank/1 and 4 times with bank/2.
  - $p(credit) = 204/2,000 = .102$
  - $p(credit \mid s_1) = 200/1,500 = .133$
  - $p(credit \mid s_2) = 4/500 = .008$
- From Bayes Rule
  - $p(s_1 \mid credit) = .133 * .75/.102 = .978$
  - $p(s_2 \mid credit) = .008 * .25/.102 = .020$
- Weights
  - $weight(s_1, credit) = \log 49.8 = 3.89$
  - $weight(s_2, credit) = \log \frac{1}{49.8} = -3.89$

Using the Decision List



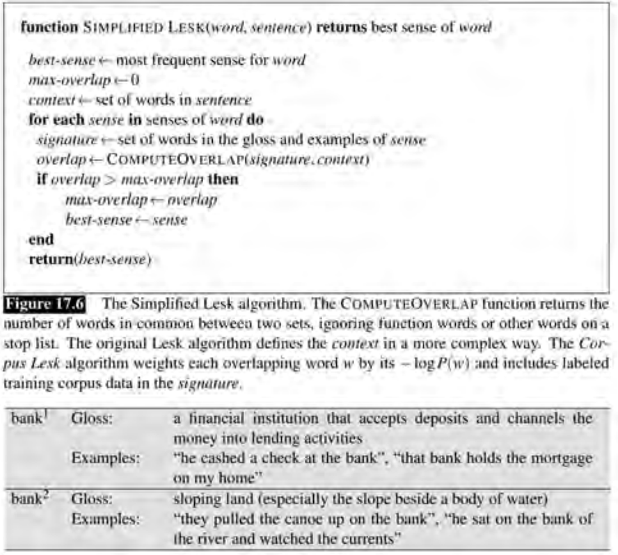
Evaluation of WSD

- Extrinsic Evaluation
  - Also called *task-based*, *end-to-end*, and *in vivo* evaluation.
  - Measures the contribution of a WSD (or other) component to a larger pipeline.
  - Requires a large investment and hard to generalize to other tasks,
- Intrinsic Evaluation
  - Also called *in vitro* evaluation
  - Measures the performance of the WSD (or other) component in isolation
  - Does not necessarily tell you how well the component contributes to a real test – which is in general what you are interested in.

Baselines

- **Most frequent sense**
  - Senses in WordNet are typically ordered from most to least frequent
  - For each word, simply pick the most frequent
  - Surprisingly accurate
- **Lesk algorithm**
  - Really, a family of algorithms
  - Measures overlap in words between gloss/examples and context

Simplified Lesk Algorithm



```
function SIMPLIFIED LESK(word, sentence) returns best sense of word

  best-sense ← most frequent sense for word
  max-overlap ← 0
  context ← set of words in sentence
  for each sense in senses of word do
    signature ← set of words in the gloss and examples of sense
    overlap ← COMPUTEOVERLAP(signature, context)
    if overlap > max-overlap then
        max-overlap ← overlap
        best-sense ← sense
  end
  return(best-sense)
```

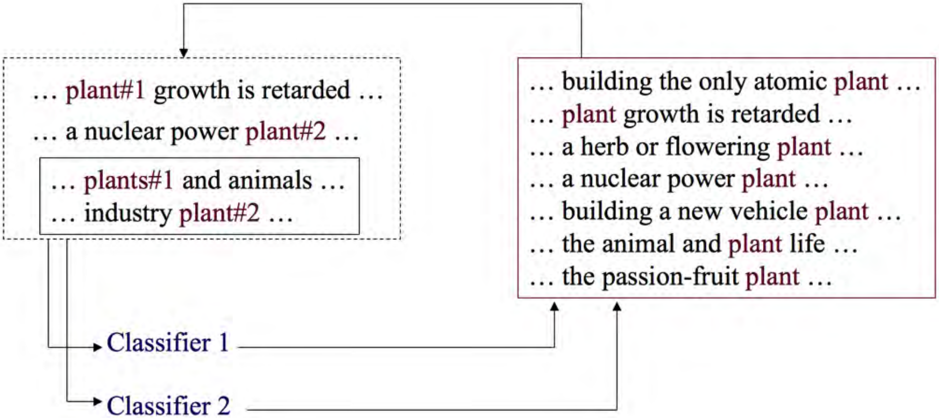
Figure 17.6 The Simplified Lesk algorithm. The COMPUTEOVERLAP function returns the number of words in common between two sets, ignoring function words or other words on a stop list. The original Lesk algorithm defines the *context* in a more complex way. The *Corpus Lesk* algorithm weights each overlapping word w by its $-\log P(w)$ and includes labeled training corpus data in the *signature*.

bank^1	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	"he cashed a check at the bank", "that bank holds the mortgage on my home"
bank^2	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	"they pulled the canoe up on the bank", "he sat on the bank of the river and watched the currents"

- The **bank** can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.
  - Sense **bank**^{1} has two non-stopwords overlapping with the context above,
  - Sense **bank**^{2} has no overlaps.

Bootstrapping Algorithms

- There are bootstrapping techniques that can be used to obtain reasonable WSD results with minimal amounts of labelled data.



… plant#1 growth is retarded …

… a nuclear power plant#2 …

… plants#1 and animals …

… industry plant#2 …

Classifier 1

Classifier 2

… building the only atomic plant …

… plant growth is retarded …

… a herb or flowering plant …

… a nuclear power plant …

… building a new vehicle plant …

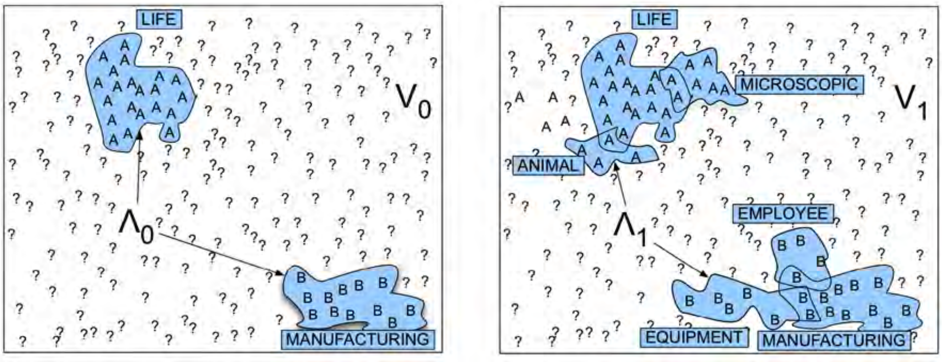
… the animal and plant life …

… the passion-fruit plant …

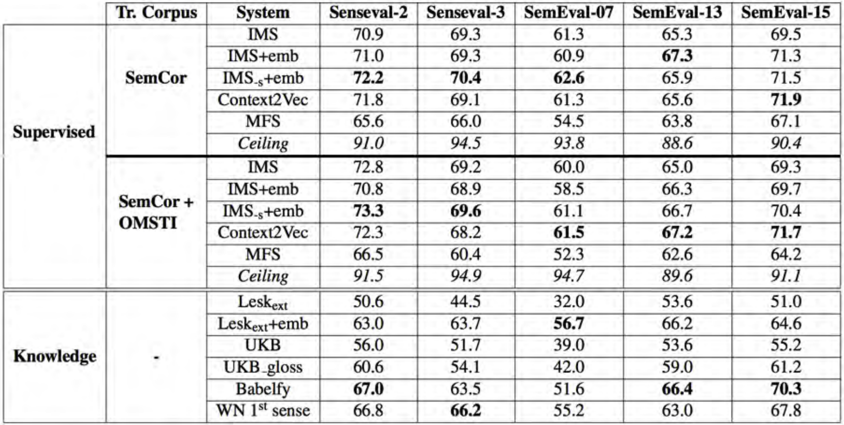
Bootstrapping Algorithms

- Yarowsky's Algorithm (1995), builds a classifier for each ambiguous word.
  - The algorithm is given a small seed set Λ_0 of labeled instances of each sense and a much larger unlabeled corpus V_0.
  - Trains an initial classifier and labels V_0 along with confidence
  - Add high-confidence labeled examples to the training set giving Λ_1
  - Trains an new classifier and labels V_1 along with confidence.
  - Add high-confidence labeled examples to the training set giving Λ_2
  - ...
  - Until no new examples can be added or a sufficiently accurate labeling is reached.
- Assumptions/Observations:
  - **One sense per collocation**: Nearby words provide strong and consistent clues as to the sense of a target word
  - **One sense per discourse**: The sense of a target word is highly consistent within a single document

Bootstrapping Example



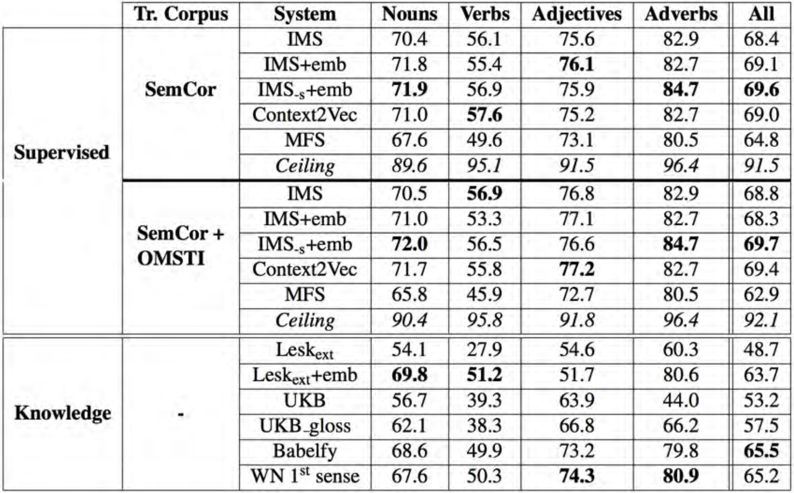
State-of-the-art Results in WSD (2017)



	Tr. Corpus	System	Senseval-2	Senseval-3	SemEval-07	SemEval-13	SemEval-15
Supervised	**SemCor**	IMS	70.9	69.3	61.3	65.3	69.5
		IMS+emb	71.0	69.3	60.9	**67.3**	71.3
		$\text{IMS}_{\text{-s}}$+emb	**72.2**	**70.4**	**62.6**	65.9	71.5
		Context2Vec	71.8	69.1	61.3	65.6	**71.9**
		MFS	65.6	66.0	54.5	63.8	67.1
		Ceiling	*91.0*	*94.5*	*93.8*	*88.6*	*90.4*
	SemCor + OMSTI	IMS	72.8	69.2	60.0	65.0	69.3
		IMS+emb	70.8	68.9	58.5	66.3	69.7
		$\text{IMS}_{\text{-s}}$+emb	**73.3**	**69.6**	61.1	66.7	70.4
		Context2Vec	72.3	68.2	**61.5**	**67.2**	**71.7**
		MFS	66.5	60.4	52.3	62.6	64.2
		Ceiling	*91.5*	*94.9*	*94.7*	*89.6*	*91.1*
Knowledge	**-**	Lesk_{ext}	50.6	44.5	32.0	53.6	51.0
		Lesk_{ext}+emb	63.0	63.7	**56.7**	66.2	64.6
		UKB	56.0	51.7	39.0	53.6	55.2
		UKB_gloss	60.6	54.1	42.0	59.0	61.2
		Babelfy	**67.0**	63.5	51.6	**66.4**	**70.3**
		WN 1^{st} sense	66.8	**66.2**	55.2	63.0	67.8

Table 2: F-Measure percentage of different models in five all-words WSD datasets.

State-of-the-art Results in WSD (2017)



	Tr. Corpus	System	Nouns	Verbs	Adjectives	Adverbs	All
Supervised	**SemCor**	IMS	70.4	56.1	75.6	82.9	68.4
		IMS+emb	71.8	55.4	**76.1**	82.7	69.1
		IMS$_{-s}$+emb	**71.9**	56.9	75.9	**84.7**	**69.6**
		Context2Vec	71.0	**57.6**	75.2	82.7	69.0
		MFS	67.6	49.6	73.1	80.5	64.8
		Ceiling	*89.6*	*95.1*	*91.5*	*96.4*	*91.5*
	SemCor + OMSTI	IMS	70.5	**56.9**	76.8	82.9	68.8
		IMS+emb	71.0	53.3	77.1	82.7	68.3
		IMS$_{-s}$+emb	**72.0**	56.5	76.6	**84.7**	**69.7**
		Context2Vec	71.7	55.8	**77.2**	82.7	69.4
		MFS	65.8	45.9	72.7	80.5	62.9
		Ceiling	*90.4*	*95.8*	*91.8*	*96.4*	*92.1*
Knowledge	-	Lesk$_{ext}$	54.1	27.9	54.6	60.3	48.7
		Lesk$_{ext}$+emb	**69.8**	**51.2**	51.7	80.6	63.7
		UKB	56.7	39.3	63.9	44.0	53.2
		UKB_gloss	62.1	38.3	66.8	66.2	57.5
		Babelfy	68.6	49.9	73.2	79.8	**65.5**
		WN 1st sense	67.6	50.3	**74.3**	**80.9**	65.2

Table 4: F-Measure percentage of different models on the concatenation of all five WSD datasets.

Other Approaches – Ensembles

- Classifier error has two components: Bias and Variance
    - The bias is error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).
    - The variance is error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).
- Some algorithms (e.g., decision trees) try and build a representation of the training data - Low Bias/High Variance
- Others (e.g., Naive Bayes) assume a parametric form and don't necessarily represent the training data - High Bias/Low Variance
- Combining classifiers with different bias variance characteristics can lead to improved overall accuracy.

Unsupervised Methods for Word Sense **Discrimination/Induction**

- Unsupervised learning identifies patterns in a large sample of data, without the benefit of any manually labeled examples or external knowledge sources.
- These patterns are used to divide the data into clusters, where each member of a cluster has more in common with the other members of its own cluster than any other.
- Important: If you remove manual labels from supervised data and cluster, you may not discover the same classes as in supervised learning:
  - Supervised Classification identifies features that trigger a sense tag
  - Unsupervised Clustering finds similarity between contexts
- Recent approaches to this use embeddings.

11-411
Natural Language Processing
Semantic Roles
Semantic Parsing

Kemal Oflazer

Carnegie Mellon University in Qatar

Semantics vs Syntax

- Syntactic theories and representations focus on the question of which strings in $\mathcal{V}^+$ are in the language.
- Semantics is about “understanding” what a string in $\mathcal{V}^+$ *means*.
- Sidestepping a lengthy and philosophical discussion of what “meaning” is, we will consider two meaning representations:
  - Predicate-argument structures, also known as event frames.
  - Truth conditions represented in first-order logic.

Motivating Example: Who did What to Whom?

- Warren bought the stock.
- They sold the stock to Warren.
- The stock was bought by Warren.
- The purchase of the stock by Warren surprised no one.
- Warren’s stock purchase surprised no one.

Motivating Example: Who did What to Whom”

- Warren bought the stock.
- They sold the stock to Warren.
- The stock was bought by Warren.
- The purchase of the stock by Warren surprised no one.
- Warren’s stock purchase surprised no one.

Motivating Example: Who did What to Whom?

- Warren bought the stock.
- They sold the stock to Warren.
- The stock was bought by Warren.
- The purchase of the stock by Warren surprised no one.
- Warren's stock purchase surprised no one.

Motivating Example: Who did What to Whom”

- Warren bought the stock.
- They sold the stock to Warren.
- The stock was bought by .

  The purchase of the stock by Warren surprised no one.

- Warren’s stock purchase surprised no one.

Motivating Example: Who did What to Whom?

- Warren bought the stock.
- They sold the stock to Warren.
- The stock was bought by Warren.
- The purchase of the stock by Warren surprised no one.
- Warren's stock purchase surprised no one.

- In this buying/purchasing event/situation, Warren played the role of the buyer, and there was some stock that played the role of the thing purchased.
- Also, there was presumably a seller, only mentioned in one example.
- In some examples, a separate "event" involving surprise did not occur.

Semantic Roles: Breaking

- Ali broke the window.
- The window broke.
- Ali is always breaking things.
- The broken window testified to Ali’s malfeasance.

Semantic Roles: Breaking

- Ali broke the window.
- The window broke. (?)
- Ali is always breaking things.
- The broken window testified to Ali's malfeasance.
- A breaking event has a BREAKER and a BREAKEE.

Semantic Roles: Eating

- Eat!
- We ate dinner.
- We already ate.
- The pies were eaten up quickly.
- Our gluttony was complete.

Semantic Roles: Eating

- Eat!(you, listener) ?
- We ate dinner.
- We already ate.
- The pies were eaten up quickly.
- Our gluttony was complete.

- A eating event has a EATER and a FOOD, neither of which needs to be mentioned explicitly.

Abstraction

$$\text{Breaker} \overset{?}{=} \text{Eater}$$

Both are actors that have some causal responsibility for changes in the world around them.

$$\text{Breakee} \overset{?}{=} \text{Food}$$

Both are greatly affected by the event, which “happened to” them.

Thematic Roles

AGENT	The waiter spilled the soup.
EXPERIENCER	Ali has a headache.
FORCE	The wind blows debris into our garden
THEME	Ali broke the window.
RESULT	The city built a basketball court.
CONTENT	Omar asked, “You saw Ali playing soccer?”
INSTRUMENT	He broke the window with a hammer.
BENEFICIARY	Jane made reservations for me.
SOURCE	I flew in from New York.
GOAL	I drove to Boston.

Verb Alternation Examples: Breaking and Giving

- Breaking:
  - AGENT/subject; THEME/object; INSTRUMENT/PP$_{\text{with}}$
  - INSTRUMENT/subject; THEME/object
  - THEME/subject
- Giving:
  - AGENT/subject; GOAL/object; THEME/second-object
  - AGENT/subject; THEME/object; GOAL/PP$_{\text{to}}$
- English verbs have been codified into classes that share patterns (e.g., verbs of throwing: throw/kick/pass)

Semantic Role Labeling

- Input: a sentence x
- Output: A collection of **predicates**, each consisting of
    - a label sometimes called the **frame**
    - a span
    - a set of **arguments**, each consisting of
        - a label usually called the **role**
        - a span

The Importance of Lexicons

- Like syntax, any annotated dataset is the product of extensive development of conventions.
- Many conventions are specific to particular words, and this information is codified in structured objects called **lexicons**.
- You should think of every semantically annotated dataset as both the data and the lexicon.
- We consider two examples.

PropBank

- Frames are *verb senses* (with some extensions)
- Lexicon maps verb-sense-specific roles onto a small set of abstract roles (e.g., ARG0, ARG1, etc.)
- Annotated on top of the Penn Treebank, so that arguments are always constituents.

fall.01 (move downward)

- ARG1: logical subject, patient, thing falling
- ARG2: extent, amount fallen
- ARG3: starting point
- ARG4: ending point
- ARGM-LOC: medium

- Sales fell to $251.2 million from $278.8 million.
- The average junk bond fell by 4.2%.
- The meteor fell through the atmosphere, crashing into Palo Alto.

fall.01 (move downward)

- ARG1: logical subject, patient, thing falling
- ARG2: extent, amount fallen
- ARG3: starting point
- ARG4: ending point
- ARGM-LOC: medium

- Sales fell to $251.2 million from $278.8 million.
- The average junk bond fell by 4.2%.
- The meteor fell through the atmosphere, crashing into Palo Alto.

fall.01 (move downward)

- ARG1: logical subject, patient, thing falling
- ARG2: extent, amount fallen
- ARG3: starting point
- ARG4: ending point
- ARGM-LOC: medium

- Sales fell to $251.2 million from $278.8 million.
- The average junk bond fell by 4.2%.
- The meteor fell through the atmosphere, crashing into Palo Alto.

fall.01 (move downward)

- ARG1: logical subject, patient, thing falling
- ARG2: extent, amount fallen
- ARG3: starting point
- ARG4: ending point
- ARGM-LOC: medium

- Sales fell to $251.2 million from $278.8 million.
- The average junk bond fell by 4.2%.
- The meteor fell through the atmosphere, crashing into Palo Alto.

fall.08 (fall back, rely on in emergency)

- ARG0: thing falling back
- ARG1: thing fallen on

- World Bank president Paul Wolfowitz has fallen back on his last resort.

fall.08 (fall back, rely on in emergency)

- ARG0: thing falling back
- ARG1: thing fallen on

- World Bank president Paul Wolfowitz has fallen back on his last resort.

fall.08 (fall back, rely on in emergency)

- ARG0: thing falling back
- ARG1: thing fallen on

- World Bank president Paul Wolfowitz has fallen back on his last resort.

fall.10 (fall for a trick; be fooled by)

- ARG0: the fool
- ARG1: the trick

- Many people keep falling for the idea that lowering taxes on the rich benefits everyone.

fall.10 (fall for a trick; be fooled by)

- ARG0: the fool
- ARG1: the trick

- Many people keep falling for the idea that lowering taxes on the rich benefits everyone.

fall.10 (fall for a trick; be fooled by)

- ARG0: the fool
- ARG1: the trick

- Many people keep falling for the idea that lowering taxes on the rich benefits everyone.

FrameNet

- Frames can be any content word (verb, noun, adjective, adverb)
- About 1,000 frames, each with its own roles
- Both frames and roles are hierarchically organized
- Annotated without syntax, so that arguments can be anything
- Different philosophy:
  - Micro roles defined according to frame
  - Verb is in the background and frame is in the foreground.
  - When a verb is “in” a frame it is allowed to use the associated roles.

change_position_on_a_scale

- ITEM: entity that has a position on the scale
- ATTRIBUTE: scalar property that the ITEM possesses
- DIFFERENCE: distance by which an ITEM changes its position
- FINAL_STATE: ITEM's state after the change
- FINAL_VALUE: position on the scale where ITEM ends up
- INITIAL_STATE: ITEM's state before the change
- INITIAL_VALUE: position on the scale from which the ITEM moves
- VALUE_RANGE: portion of the scale along which values of ATTRIBUTE fluctuate
- DURATION: length of time over which the change occurs
- SPEED: rate of change of the value

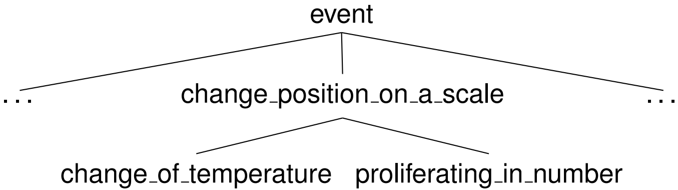
FrameNet Example

$$\underbrace{\text{Attacks on civilians}}_{\textsc{Item}} \quad \underbrace{\text{decreased}}_{\text{change\_position...}} \quad \underbrace{\text{over the last four months}}_{\textsc{Duration}}$$

- The ATTRIBUTE is left unfilled but is understood from context (e.g., “number” or “frequency”).

change_position_on_a_scale

- *Verbs*: advance, climb, decline, decrease, diminish, dip, double, drop, dwindle, edge, explode, fall, fluctuate, gain, grow, increase, jump, move, mushroom, plummet, reach, rise, rocket, shift, skyrocket, slide, soar, swell, swing, triple, tumble
- *Nouns*: decline, decrease, escalation, explosion, fall, fluctuation, gain, growth, hike, increase, rise, shift, tumble
- Adverb: increasingly
- Frame hierarchy



event

. . .

change_position_on_a_scale

. . .

change_of_temperature

proliferating_in_number

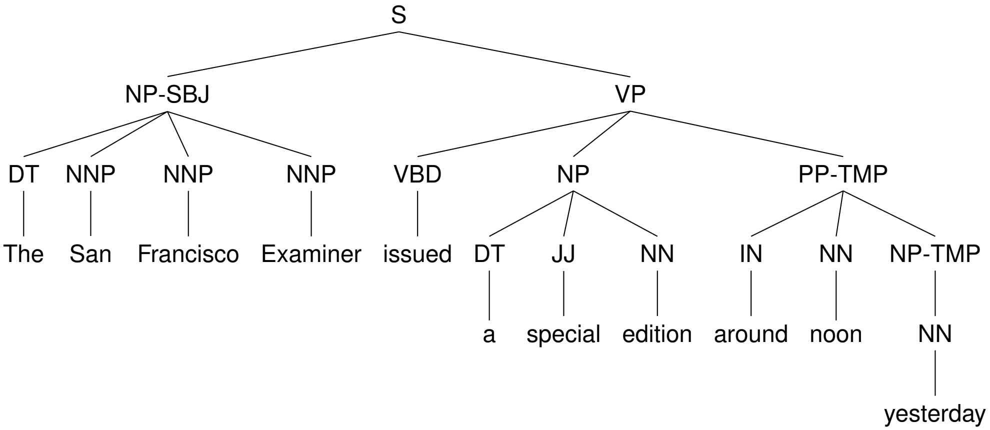
The Semantic Role Labeling Task

- Given a syntactic parse, identify the appropriate role for each noun phrase (according to the scheme that you are using, e.g., PropBank, FrameNet or something else)
- Why is this useful?
  - Why is it useful for some tasks that you cannot perform with just dependency parsing?
  - What kind of semantic representation could you obtain if you had SRL?
- Why is this hard?
  - Why is it harder that dependency parsing?

Semantic Role Labeling Methods

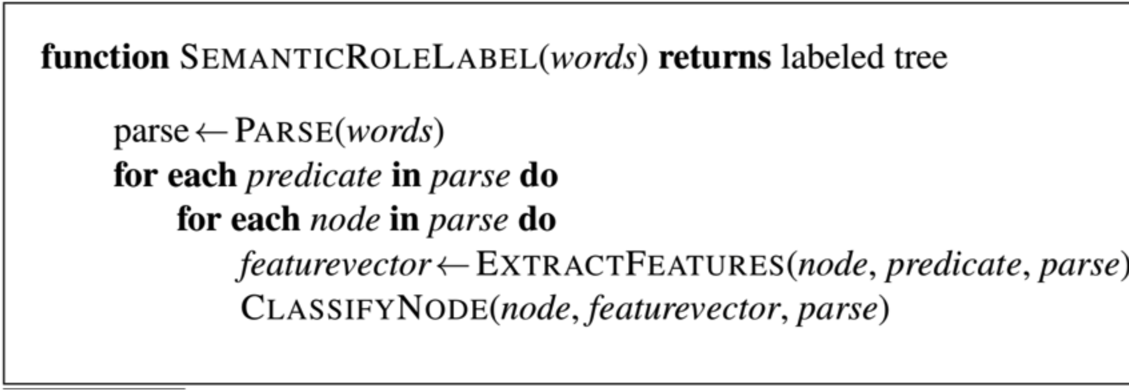
- Boils down to labeling spans with frames and role names.
- **It is mostly about features.**
- Some features for SRL
  - The governing predicate (often the main verb)
  - The phrase type of the constituent (NP, NP-SUBJ, etc)
  - The headword of the constituent
  - The part of speech of the headword
  - The path from the constituent to the predicate
  - The voice of the clause (active, passive, etc.)
  - The binary linear position of the constituent with respect to the predicate (before or after)
  - The subcategorization of the predicate
  - Others: named entity tags, more complex path features, when particular nodes appear in the path, rightmost and leftmost words in the constituent, etc.

Example: Path Features



Path from “The San Francisco Examiner” to “issued”: NP↑S↓VP↓VBD
Path from “a special edition” to “issued”: NP↑VP↓VBD

Sketch of an SRL Algorithm



```
function SEMANTICROLELABEL(words) returns labeled tree

    parse ← PARSE(words)
    for each predicate in parse do
        for each node in parse do
            featurevector ← EXTRACTFEATURES(node, predicate, parse)
            CLASSIFYNODE(node, featurevector, parse)
```

Additional Steps for Efficiency

- **Pruning**
  - Only a small number of constituents should ultimately be labeled
  - Use heuristics to eliminate some constituents from consideration
- **Preliminary Identification**:
  - Label each node as ARG or NONE with a binary classifier
- **Classification**
  - Only then, perform 1-of-N classification to label the remaining ARG nodes with roles

Additional Information

- See `framenet/icsi.berkeley.edu/fndrupal/` for additional information about the FrameNet Project.
- Semantic Parsing Demos at
  - `http://demo.ark.cs.cmu.edu/parse`
  - `http://nlp.cs.lth.se/demonstrations/`

Methods: Beyond Features

- The span-labeling decisions interact a lot!
- Presence of a frame increases the expectation of certain roles
- Roles for the same predicate should not overlap
- Some roles are mutually exclusive or require each other (e.g., “resemble”)
- Using syntax as a scaffold allows efficient prediction; you are essentially labeling the parse tree.
- Other methods include: discrete optimization, greedy and joint syntactic and semantic dependencies.

Related Problems in “Relational” Semantics

- **Coreference resolution**: which mentions (within or across texts) refer to the same entity or event?
- **Entity linking**: ground such mentions in a structured knowledge base (e.g., Wikipedia)
- **Relation extraction**: characterize the relation among specific mentions

General Remarks

- Semantic roles are just “syntax++” since they don’t allow much in the way of reasoning (e.g., question answering).
- Lexicon building is slow and requires expensive expertise. Can we do this for every (sub)language?

Snapshot

- We have had a taste of two branches of semantics:
  - Lexical semantics (e.g., supersense tagging, WSD)
  - Relational semantics (e.g., semantic role labeling)
- Coming up:
  - Compositional Semantics

11-411
Natural Language Processing
Compositional Semantics

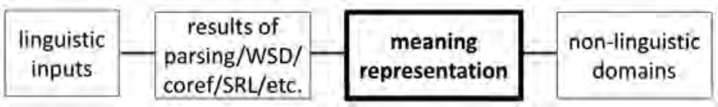
Kemal Oflazer

Carnegie Mellon University in Qatar

Semantics Road Map

- Lexical semantics
- Vector semantics
- Semantic roles, semantic parsing
- **Meaning representation languages and Compositional semantics**
- Discourse and pragmatics

Bridging the Gap between Language and the World



- Meaning representation is the interface between the language and the world.
  - Answering essay question on an exam.
  - Deciding what to order at a restaurant.
  - Recognizing a joke.
  - Executing a command.
  - Responding to a request.

Desirable Qualities of Meaning Representation Languages (MRL)

- Represent the state of the world, i.e., be a knowledge base
- Query the knowledge base to verify that a statement is true, or answers a question.
  - “Does Bukhara serve vegetarian food?”
  - Is *serves(Bukhara, vegetarian food)* in our knowledge base?
- Handle ambiguity, vagueness, and non-canonical forms
  - “I want to eat someplace that’s close to the campus.”
  - “something not too spicy”
- Support inference and reasoning.
  - “Can vegetarians eat at Bukhara?”

Desirable Qualities of Meaning Representation Languages (MRL)

- Inputs that mean the same thing should have the same meaning representation.
  - “Bukhara has vegetarian dishes.”
  - “They have vegetarian food at Bukhara.”
  - “Vegetarian dishes are served at Bukhara.”
  - “Bukhara serves vegetarian fare.”

Variables and Expressiveness

- “ I would like to find a restaurant where I can get vegetarian food.”
  - `serves(x, vegetarian food)`
- It should be possible to express all the relevant details
  - “Qatar Airways flies Boeing 777s and Airbus 350s from Doha to the US”

Limitation

- We will focus on the basic requirements of meaning representation.
- These requirements do not include correctly interpreting statements like
  - “Ford was hemorrhaging money.”
  - “I could eat a horse.”

What do we Represent?

- **Objects**: people (John, Ali, Omar), cuisines (Thai, Indian), restaurants (Bukhara, Chef's Garden), . . .
  - *John, Ali, Omar, Thai, Indian, Chinese, Bukhara, Chefs Garden, . . .*
- **Properties of Objects**: Ali is picky, Bukhara is noisy, Bukhara is cheap, Indian is spicy, John, Ali and Omar are humans, Bukhara has long wait . . .
  - *picky={Ali}, noisy={Bukhara}, spicy={Indian}, human={Ali, John, Omar}. . .*
- **Relations between objects**: Bukhara serves Indian, NY Steakhouse serves steak. Omar likes Chinese.
  - *serves(Bukhara, Indian), serves(NY Steakhouse, steak), likes(Omar, Chinese) . . .*
- Simple questions are easy:
  - Is Bukhara noisy?
  - Does Bukhara serve Chinese?

MRL: First-order Logic – A Quick Tour

- **Term**: any constant (e.g., *Bukhara*) or a variable
- **Formula**: defined inductively . . .
  - if R is an n-ary relation and $t_1, \ldots, t_n$ are terms, then $R(t_1, \ldots, t_n)$ is a formula.
  - if ϕ is a formula, then its negation, $\neg\phi$ is a formula.
  - if ϕ and ψ are formulas, then *binary logical connectives* can be used to create formulas:
    - $\phi \wedge \psi$
    - $\phi \vee \psi$
    - $\phi \Rightarrow \psi$
    - $\phi \oplus \psi$
  - If ϕ is a formula and v is a variable, then *quantifiers* can be used to create formulas:
    - Existential quantifier: $\exists v : \phi$
    - Universal quantifier: $\forall v : \phi$

First-order Logic: Meta Theory

- Well-defined set-theoretic semantics
- **Sound**: You can't prove false things.
- **Complete**: You can prove everything that logically follows from a set of axioms (e.g. with a "resolution theorem prover.")
- Well-behaved, well-understood.
- But there are issues:
  - "Meanings" of sentences are truth values.
  - Only **first-order** (no quantifying over predicates).
  - Not very good for "fluents" (time-varying things, real-valued quantities, etc.)
  - Brittle: *anything* follows from any contradiction (!)
  - **Gödel Incompleteness**: "This statement has no proof."
    - Finite axiom sets are incomplete with respect to the real world.
- Most systems use its descriptive apparatus (with extensions) but not its inference mechanisms.

Translating between First-order Logic and Natural Language

- Bukhara is not loud. $\neg noisy(Bukhara)$
- Some humans like Chinese. $\exists x,\ human(x) \land likes(x, chinese)$
- If a person likes Thai, then they are not friends with Ali. $\forall x,\ human(x) \land likes(x, Thai) \Rightarrow \neg friends(x, Ali)$
- $\forall x, restaurant(x) \Rightarrow (longwait(x) \lor \neg likes(Ali, x))$ Every restaurant has a long wait or is disliked by Ali.
- $\forall x, \exists y, \neg likes(x, y)$ Everybody has something they don't like.
- $\exists y, \forall x, \neg likes(x, y)$ There is something that nobody likes.

Logical Semantics (Montague Semantics)

- The *denotation* of a natural language sentence is the set of conditions that must hold in the (model) world for the sentence to be true.
- “Every restaurant has a long wait or is disliked by Ali.”
  is true if an only if

  $$\forall x, restaurant(x) \Rightarrow (longwait(x) \vee \neg likes(Ali, x))$$

  is true.
- This is sometimes called the **logical form** of the NL sentence.

The Principle of Compositionality

- The meaning of a natural language phrase is determined by the meanings of its sub-phrases.
    - There are obvious exceptions: e.g., hot dog, New York, etc.
- Semantics is derived from syntax.
- We need a way to express semantics of phrases, and compose them together!
- Little pieces of semantics are **introduced by words**, from the lexicon.
- Grammar rules include **semantic attachments** that describe how the semantics of the children are **combined** to produce the semantics of the parent, bottom-up.

Lexicon Entries

- In real systems that do detailed semantics, lexicon entries contain
  - Semantic attachments
  - Morphological info
  - Grammatical info (POS, etc.)
  - Phonetic info, if speech system
  - Additional comments, etc.

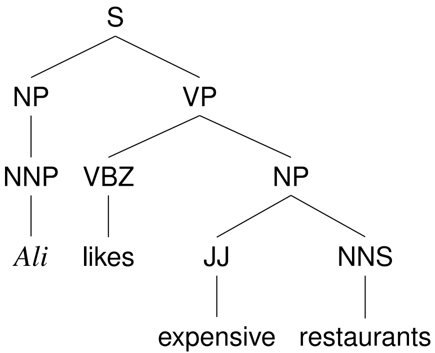
λ-Calculus

- λ-**abstraction** is device way to give “scope” to variables.
  - If ϕ is a formula and v is a variable, then $\lambda v.\phi$ is a λ-term: an unnamed function from values (of v) to formulas (usually involving v)
- **application** of such functions: if we have $\lambda v.\phi$ and ψ, then $[\lambda v.\phi](\psi)$ is a formula.
  - It can be reduced by substituting ψ for every instance of v in ϕ
  - $[\lambda x.likes(x, Bukhara)](Ali)$ reduces to $likes(Ali, Bukhara)$.
  - $[\lambda x.\lambda y.friends(x, y)](b)$ reduces to $\lambda y.friends(b, y)$
  - $[[\lambda x.\lambda y.friends(x, y)](b)](a)$ reduces to $[\lambda y.friends(b, y)](a)$ which reduces to $friends(b, a)$

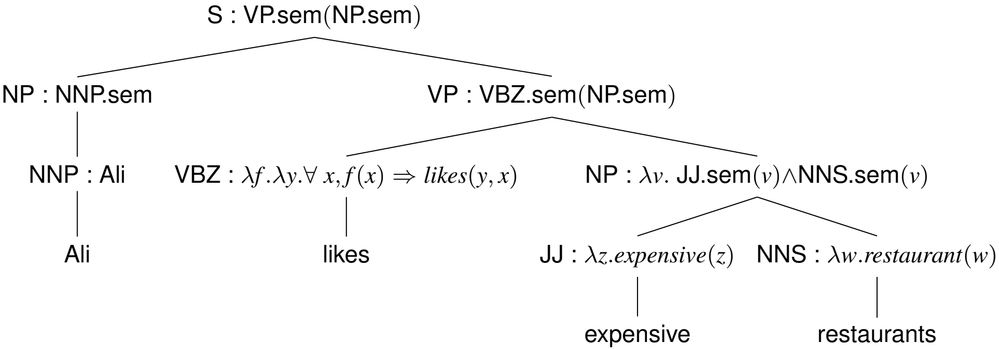
Semantic Attachments to CFGs

- NNP → Ali $\{Ali\}$
- VBZ → likes $\{\lambda f.\lambda y.\forall x\, f(x) \Rightarrow likes(y,x)\}$
- JJ → expensive $\{\lambda x.expensive(x)\}$
- NNS → restaurants $\{\lambda x.restaurant(x)\}$
- NP → NNP $\{\text{NNP.sem}\}$
- NP → JJ NNS $\{\lambda x.\text{JJ.sem}(x) \wedge \text{NNS.sem}(x)\}$
- VP → VBZ NP $\{\text{VBZ.sem}(\text{NP.sem})\}$
- S → NP VP $\{\text{VP.sem}(\text{NP.sem})\}$

Example



Example



S : VP.sem(NP.sem)

NP : NNP.sem

VP : VBZ.sem(NP.sem)

NNP : Ali

VBZ : $\lambda f.\lambda y.\forall\, x, f(x) \Rightarrow likes(y, x)$

NP : $\lambda v.$ JJ.sem$(v)\wedge$NNS.sem(v)

Ali

likes

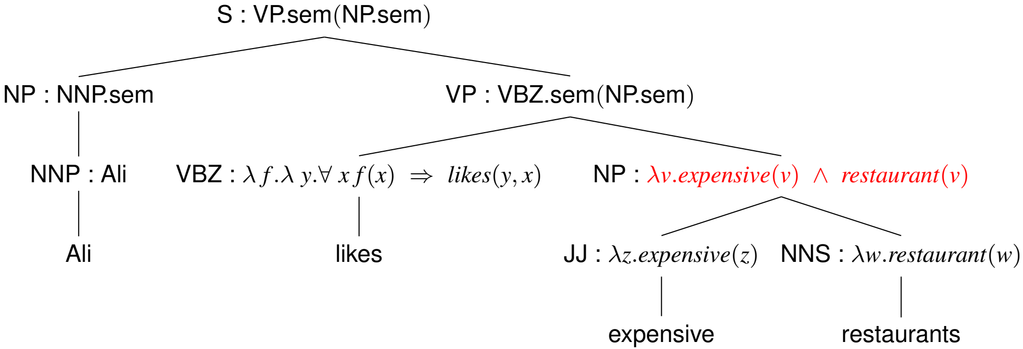
JJ : $\lambda z.expensive(z)$

NNS : $\lambda w.restaurant(w)$

expensive

restaurants

Example



S : VP.sem(NP.sem)

NP : NNP.sem

VP : VBZ.sem(NP.sem)

NNP : Ali

Ali

VBZ : $\lambda f.\lambda y.\forall x\, f(x) \Rightarrow likes(y,x)$

likes

NP : $\lambda v.expensive(v) \wedge restaurant(v)$

JJ : $\lambda z.expensive(z)$

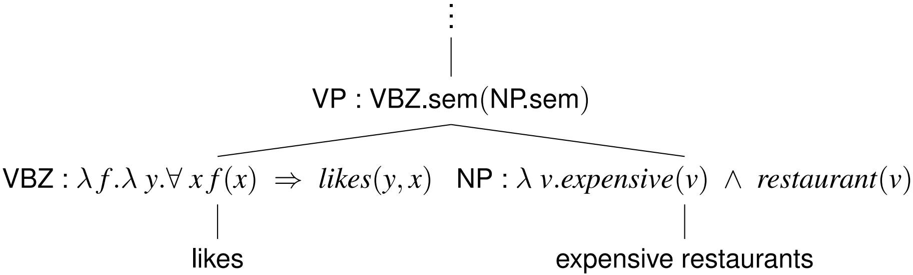
expensive

NNS : $\lambda w.restaurant(w)$

restaurants

$$\lambda v. \left[\underbrace{\lambda z.expensive(z)}_{\text{JJ.sem}}\right](v) \wedge \left[\underbrace{\lambda w.restaurant(w)}_{\text{NNS.sem}}\right](v)$$

Example



⋮

VP : VBZ.sem(NP.sem)

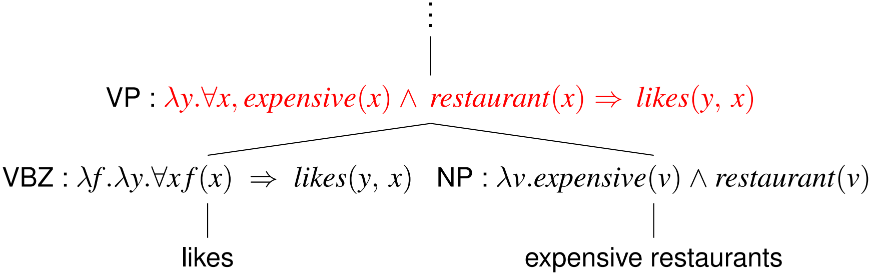
VBZ : $\lambda f.\lambda y.\forall x\, f(x) \Rightarrow likes(y,x)$

NP : $\lambda v.expensive(v) \wedge restaurant(v)$

likes

expensive restaurants

Example



⋮

VP : $\lambda y.\forall x, expensive(x) \wedge restaurant(x) \Rightarrow likes(y, x)$

VBZ : $\lambda f.\lambda y.\forall x f(x) \Rightarrow likes(y, x)$

NP : $\lambda v.expensive(v) \wedge restaurant(v)$

likes

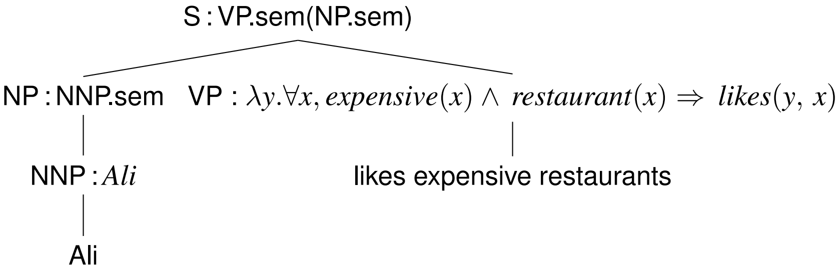
expensive restaurants

$$\left[\underbrace{\lambda f.\lambda y.\forall x f(x) \Rightarrow likes(y, x)}_{\text{VBZ.sem}}\right]\left(\underbrace{\lambda v.expensive(v) \wedge restaurant(v)}_{\text{NP.sem}}\right)$$

$$\lambda y.\forall x\, [\lambda v.expensive(v) \wedge restaurant(v)](x) \Rightarrow likes(y, x)$$

$$\lambda y.\forall x,\ expensive(x) \wedge restaurant(x) \Rightarrow likes(y, x)$$

Example



S : VP.sem(NP.sem)

NP : NNP.sem

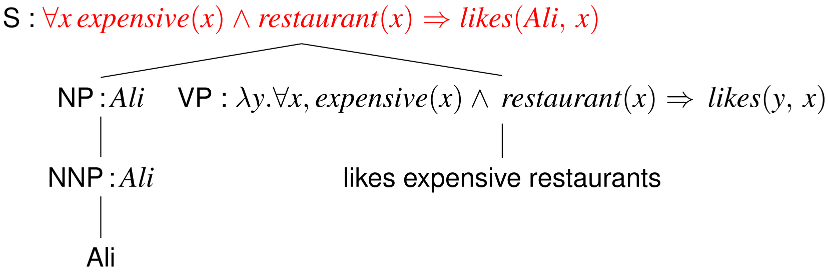
VP : $\lambda y.\forall x, expensive(x) \wedge restaurant(x) \Rightarrow likes(y, x)$

NNP : Ali

likes expensive restaurants

Ali

Example



S : $\forall x\, expensive(x) \wedge restaurant(x) \Rightarrow likes(Ali, x)$

NP : Ali

VP : $\lambda y.\forall x, expensive(x) \wedge restaurant(x) \Rightarrow likes(y, x)$

NNP : Ali

likes expensive restaurants

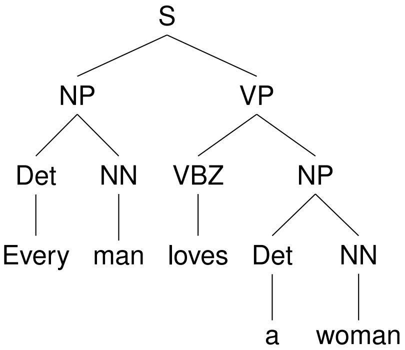
Ali

$$\left[\underbrace{\lambda y.\forall x\, expensive(x) \wedge restaurant(x) \Rightarrow likes(y, x)}_{\text{VP.sem}}\right]\left(\underbrace{Ali}_{\text{NP.sem}}\right)$$

$$\forall x\, expensive(x) \wedge restaurant(x) \Rightarrow likes(Ali, x)$$

Quantifier Scope Ambiguity

- NNP → Ali $\{Ali\}$
- VBZ → likes $\{\lambda f.\lambda y.\forall x\, f(x) \Rightarrow likes(y,x)\}$
- JJ → expensive $\{\lambda x.expensive(x)\}$
- NNS → restaurants $\{\lambda x.restaurant(x)\}$
- NP → NNP $\{\text{NNP.sem}\}$
- NP → JJ NNS $\{\lambda x.\text{JJ.sem}(x) \wedge \text{NNS.sem}(x)\}$
- VP → VBZ NP $\{\text{VBZ.sem}(\text{NP.sem})\}$
- S → NP VP $\{\text{VP.sem}(\text{NP.sem})\}$
- NP → Det NN $\{\text{Det.sem}(\text{NN.sem})\}$
- Det → every $\{\lambda f.\lambda g.\forall u\, f(u) \Rightarrow g(u)\}$
- Det → a $\{\lambda m.\lambda n.\exists x\, m(x) \Rightarrow n(x)\}$
- NN → man $\{\lambda v.man(v)\}$
- NN → woman $\{\lambda v.woman(v)\}$
- VBZ → loves $\{\lambda f.\lambda y.\forall x\, f(x) \Rightarrow loves(y,x)\}$

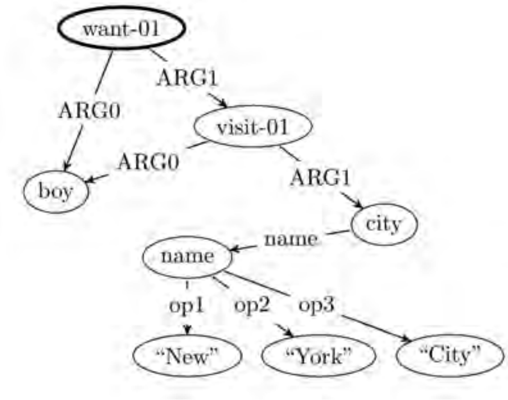


$$\forall u\, man(u) \Rightarrow \exists x\, woman(x) \wedge loves(u,x)$$

This is not Quite Right!

- "Every man loves a woman." really is ambiguous.
  - $\forall u\,(man(u) \Rightarrow \exists x\, woman(x) \wedge loves(u,x))$
  - $\exists x\,(woman(x) \wedge \forall u\, man(u) \Rightarrow loves(u,x))$
- We get only one of the two meanings.
  - Extra ambiguity on top of syntactic ambiguity.
- One approach is to delay the quantifier processing until the end, then permit any ordering.

Other Meaning Representations: Abstract Meaning Representation



- "The boy wants to visit New York City."
- Designed mainly for annotation-ability and eventual use in machine translation.

Combinatory Categorial Grammar

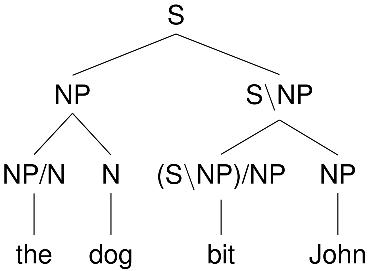
- CCG is a grammatical formalism that is well-suited for tying together syntax and semantics.
- Formally, it is more powerful than CFG – it can represent some of the context-sensitive languages.
- Instead of the set of non-terminals of CFGs, CCGs can have an infinitely large set of structured categories (called **types**).

CCG Types and Combinators

- **Primitive types**: typically S, NP, N, and maybe more.
- **Complex types**: built with “slashes,” for example:
  - S/NP is “an S, except it lacks an NP to the right”
  - S\NP is “an S, except it lacks an NP to the left”
  - (S\NP)/NP is “an S, except that it lacks an NP to its right and to its left”
- You can think of complex types as functions:
  - S/NP maps NPs to Ss.
- **CCG Combinators**: Instead of the production rules of CFGs, CCGs have a very small set of generic combinators that tell us how we can put types together.
- Convention writes the rule differently from CFG: $XY \Rightarrow Z$ means that X and Y combine to form a Z (the “parent” in the tree).

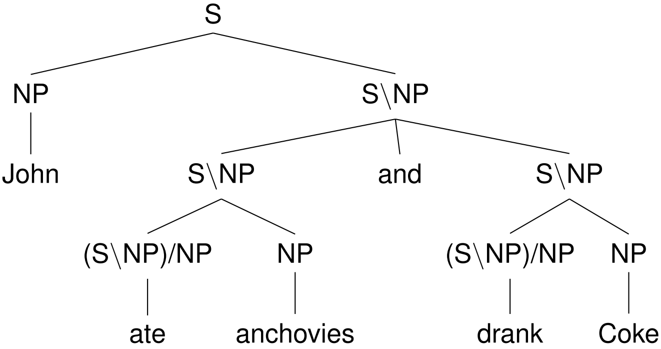
Application Combinator

- Forward Combination: $X/Y \quad Y \Rightarrow X$
- Backward Combination: $Y \quad X\backslash Y \Rightarrow X$



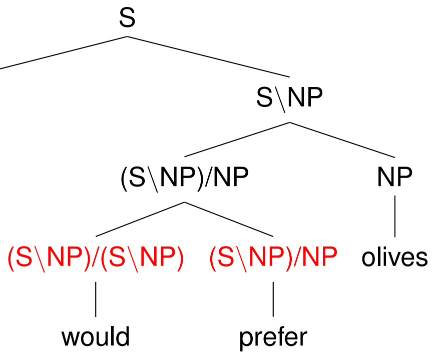
Conjunction Combinator

- X and $X \Rightarrow X$



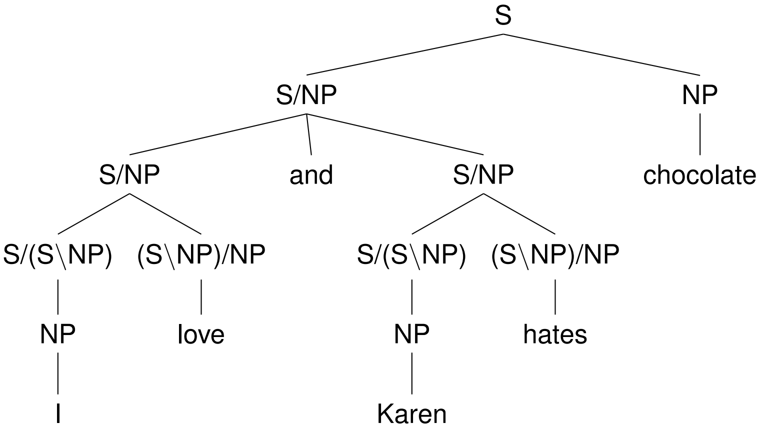
Composition Combinator

- Forward $(X/Y \quad Y/Z \Rightarrow X/Z)$
- Backward $(Y\backslash Z \quad X\backslash Y \Rightarrow X\backslash Z)$



Type-raising Combinator

- Forward $(X \Rightarrow Y/(Y\backslash X))$
- Backward $(X \Rightarrow Y\backslash(Y/X))$



Back to Semantics

- Each combinator also tells us what to do with the semantic attachments.
  - Forward application: $X/Y \; : f \quad Y \; : g \Rightarrow X \;\; f(g)$
  - Forward composition: $X/Y \; : f \quad Y/Z \; : g \Rightarrow X/Z \;\; \lambda x.f(g(x))$
  - Forward type-raising: $X \;\; : \;\; g \Rightarrow Y/(Y\backslash X) \;\; : \;\; \lambda f.f(g)$

CCG Lexicon

- Most of the work is done in the lexicon.
- Syntactic and semantic information is much more formal here.
- Slash categories define where all the syntactic arguments are expected to be
- λ-expressions define how the expected arguments get “used” to build up a FOL expression.

11-411
Natural Language Processing
Discourse and Pragmatics

Kemal Oflazer

Carnegie Mellon University in Qatar

What is Discourse?

- **Discourse** is the coherent structure of language above the level of sentences or clauses.
- A **discourse** is a coherent structured group of sentences.
- What makes a passage coherent?
  - A practical answer: It has meaningful connections between its utterances.

Applications of Computational Discourse

- Automatic essay grading
- Automatic summarization
- Meeting understanding
- Dialogue systems

Kinds of Discourse Analysis

- Monologue
- Human-human dialogue (conversation)
- Human-computer dialogue (conversational agents)
- “Longer-range” analysis (discourse) vs. “deeper” analysis (real semantics):
  - John bought a car from Bill.
  - Bill sold a car to John.
  - They were both happy with the transaction.

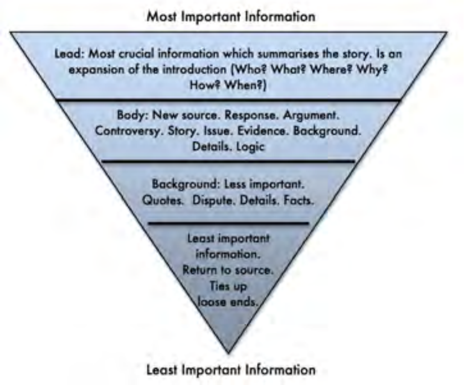
Discourse in NLP

Coherence

- Coherence relations: EXPLANATION, CAUSE
  - John hid Bill's car keys. He was drunk.
  - John hid Bill's car keys. He likes spinach.
- Consider:
  - John went to the store to buy a piano.
  - He had gone to the store for many years.
  - He was excited that he could finally afford a piano.
  - He arrived just as the store was closing for the day.
- Now consider this:
  - John went to the store to buy a piano.
  - It was a store he had gone to for many years.
  - He was excited that he could finally afford a piano.
  - It was closing for the day just as John arrived.
- First is "intuitively" more coherent than the second.
- Entity-based coherence (centering).

Discourse Segmentation

- Many genres of text have particular conventional structures:
  - Academic articles: *Abstract*, *Introduction*, *Methodology*, *Results*, *Conclusion*, etc.
  - Newspaper stories:



Most Important Information

Lead: Most crucial information which summarises the story. Is an expansion of the introduction (Who? What? Where? Why? How? When?)

Body: New source. Response. Argument. Controversy. Story. Issue. Evidence. Background. Details. Logic

Background: Less important. Quotes. Dispute. Details. Facts.

Least important information. Return to source. Ties up loose ends.

Least Important Information

- Spoken patient reports by doctors (SOAP): *Subjective*, *Objective*, *Assesment*, *Plan*.

Discourse Segmentation

- Given raw text, separate a document into a linear sequence of subtopics.

TECHNOLOGY

STARGAZERS LOOK FOR LIFE

⇒

1–3	Intro: The search for life in space
4–5	The moon's chemical composition
6–8	How early earth-moon proximity shaped the moon
9–12	How the moon helped life evolve on earth
13	Improbability of the earth-moon system
14–16	Binary/trinary star systems make life unlikely
17–18	The low probability of nonbinary/trinary systems
19–20	Properties of earth's sun that facilitate life
21	Summary

Applications of Discourse Segmentation

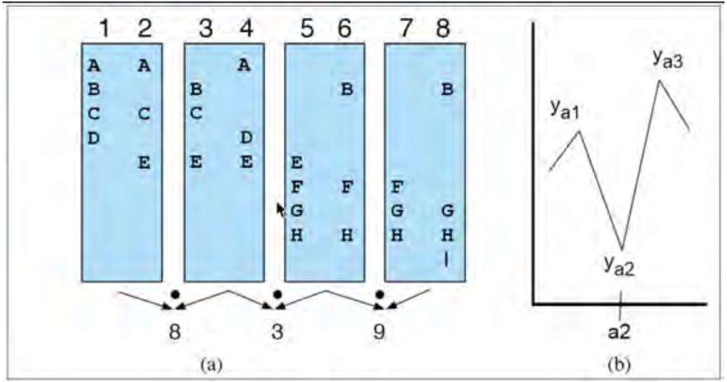
- Summarization: Summarize each segment independently.
- News Transcription: Separate a steady stream of transcribed news to separate stories.
- Information Extraction: First identify the relevant segment and then extract.

Cohesion

- To remind: **Coherence** refers to the "meaning" relation between two units. A coherence relation explains how the meaning in different textual units can combine to a meaningful discourse.
- On the other hand, **cohesion** is the use of linguistic devices to link or tie together textual units. A cohesive relation is like a "glue" grouping two units into one.
- Common words used are cues for cohesion.
  - Before winter, **I** built a chimney and **shingled** the sides of my **house**.
  - **I** have thus a tight **shingled** and plastered **house**.
- Synonymy/hypernymy relations are cues for **lexical cohesion**.
  - Peel, core and slice **the pears and the apples**.
  - Add **the fruit** to the skillet.
- Use of **anaphora** are cues for lexical cohesion
  - **The Woodhouses** were first in consequence there.
  - All looked up to **them.**

Discourse Segmentation

- **Intuition**: If we can “measure” the cohesion between every neighboring pair of sentences, we may expect a “dip” in cohesion at subtopic boundaries.
- The *TextTiling* algorithm uses lexical cohesion.



The TextTiling Algorithm

- Tokenization
  - lowercase, remove stop words, morphologically stem inflected words
  - stemmed words are (dynamically) grouped into *pseudo-sentences* of length 20 (equal length and not real sentences!)
- Lexical score determination
- Boundary identification

TextTiling – Determining Lexical Cohesion Scores

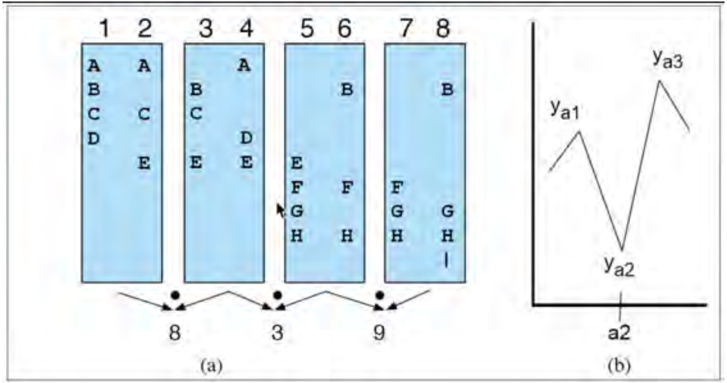
- Remember:
  - Count-based similarity vectors
  - Cosine-similarity

$$sim_{cosine}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{|\boldsymbol{a}|\,|\boldsymbol{b}|}$$

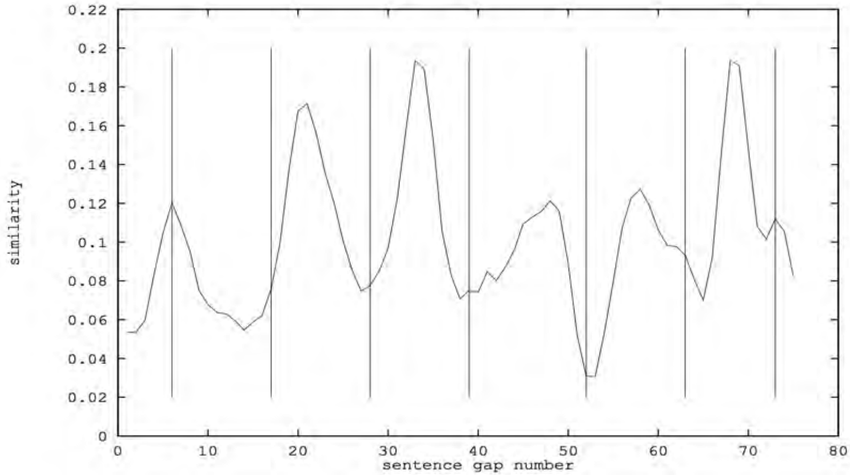
- Consider a gap position i between any two words.
- Consider $k = 10$ words before the gap ($\boldsymbol{a}$) and 10 words after the gap ($\boldsymbol{b}$), and compute their similarity y_i.
- So y_i's are the lexical cohesion scores between the 10 words before the gap and 10 words after the gap.

TextTiling – Determining Boundaries

- A gap position i is a valley if $y_i < y_{i-1}$ and $y_i < y_{i+1}$.
- If i is a valley, find the depth score – distance from the peaks on both sides $= (y_{i-1} - y_i) + (y_{i+1} - y_i)$.
- Any valley with depth at $\overline{s} - \sigma_s$ or lower, that is, deeper than one standard deviation from average valley depth, is selected as a boundary.



TextTiling – Determining Boundaries



TextTiling – Determining Boundaries

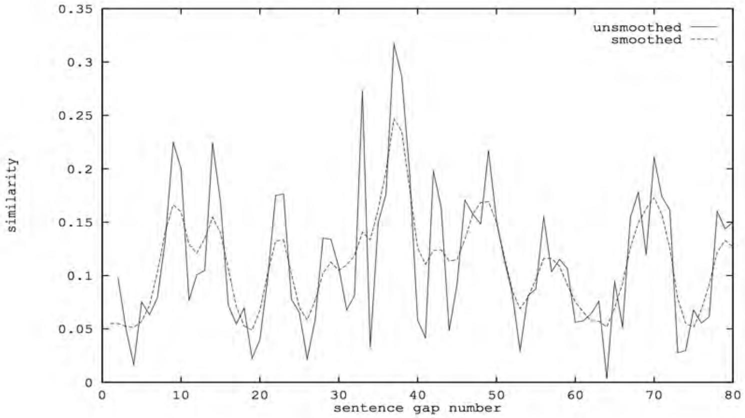


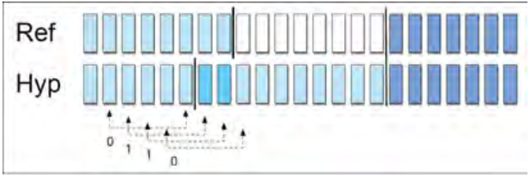
Figure 2: Smoothed and unsmoothed analyses of "Earth" (before median smoothing).

Supervised Discourse Segmentation

- Spoken news transcription segmentation task.
  - Data sets with hand-labeled boundaries exist.
- Paragraph segmentation task of monologues (lectures/speeches).
  - Plenty of data on the web with <p> markers.
- Treat the task as a *binary decision classification* problem.
- Use classifiers such as decision trees, Support Vector Machines to classify boundaries.
- Use sequence models such as Hidden Markov Models or Conditional Random Fields to incorporate sequential constraints.
- Additional features that could be used are **discourse markers** or **cue words** which are typically domain dependent.
  - *Good Evening I am PERSON*
  - *Joining us with the details is PERSON*
  - *Coming up*

Evaluating Discourse Segmentation

- We could do precision, recall and F-measure, but . . .
- These will not be sensitive to *near misses*!
- A commonly-used metric is *WindowDiff*.
- Slide a window of length k across the (correct) references and the hypothesized segmentation.



- Count the number of segmentation boundaries in each.
- Compute the *average* difference in the number of boundaries in the sliding window.
- Assigns *partial credit*.
- Another metric is $p_k(ref, hyp)$ – the probability that a randomly chosen pair of words a distance k words apart are inconsistently classified.

Coherence

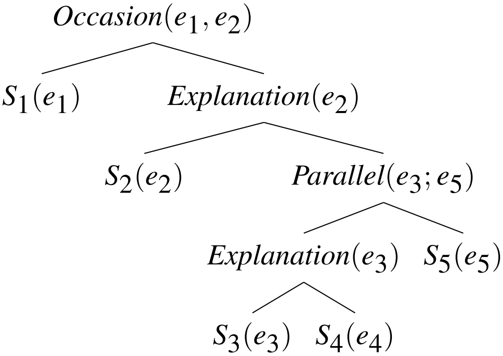
- Cohesion does not necessarily imply coherence.
- We need a more detailed definition of coherence.
- We need computational mechanisms for determining coherence.

Coherence Relations

- Let S_0 and S_1 represent the “meanings” of two sentences being related.
- **Result**: Infer that state or event asserted by S_0 causes or could cause the state or event asserted by S_1.
  - *The Tin Woodman was caught in the rain. His joints rusted.*
- **Explanation**: Infer that state or event asserted by S_1 causes or could cause the state or event asserted by S_0.
  - *John hid the car’s keys. He was drunk.*
- **Parallel**: Infer $p(a_1, a_2, \ldots)$ from the assertion of S_0 and $p(b_1, b_2, \ldots)$ from the assertion of S_1, where a_i and b_i are similar for all i.
  - *The Scarecrow wanted some brains. The Tin Woodman wanted a heart.*
- **Elaboration**: Infer the same proposition from the assertions S_0 and S_1.
  - *Dorothy was from Kansas. She lived in the midst of the great Kansas prairies.*
- **Occasion**:
  - A change of state can be inferred from the assertion S_0 whose *final* state can be inferred from S_1, or
  - A change of state can be inferred from the assertion S_1 whose *initial* state can be inferred from S_0.
  - *Dorothy picked up the oil can. She oiled the Tin Woodman’s joints.*

Coherence Relations

- Consider
  - S1: John went to the bank to deposit his paycheck.
  - S2: He then took a bus to Bill’s car dealership.
  - S3: He needed to buy a car.
  - S4: The company he works for now isn’t near a bus line.
  - S5: He also wanted to talk with Bill about their soccer league.



Rhetorical Structure Theory – RST

- Based on 23 *rhetorical relations* between two spans of text in a discourse.
  - a **nucleus** – central to the write's purpose and interpretable independently
  - a **satellite** – less central and generally only interpretable with respect to the nucleus
- Evidence relation: $\underbrace{\text{Kevin must be here.}}_{nucleus}$ $\underbrace{\text{His car is parked outsize.}}_{satellite}$
- An RST relation is defined by a set of constraints.

Relation name:	Evidence
Constraints on Nucleus	Reader might not believe Nucleus to a degree satisfactory to Writer
Constraints on Satellite	Reader believes Satellite or will find it credible.
Constraints on Nucleus+Satellite	Reader's comprehending Satellite increases Reader's belief of Nucleus
Effects	Reader's belief on Nucleus is increased.

RST – Other Common Relations

- **Elaboration**: Satelite gives more information about the nucleus
  - $[_N$ The company wouldn't elaborate,$]$ $[_S$ citing competitive reasons.$]$
- **Attribution**: The satellite gives the source of attribution for the information in nucleus.
  - $[_S$ Analysts estimated,$]$ $[_N$ that sales at US stores declined in the quarter.$]$
- **Contrast**: Two of more nuclei contrast along some dimension.
  - $[_N$ The man was in a bad temper,$]$ $[_N$ but his dog was quite happy.$]$
- **List**: A series of nuclei are given without contrast or explicit comparison.
  - $[_N$ John was the goalie;$]$ $[_N$ Bob, he was the center forward.$]$
- **Background**: The satellite gives context for interpreting the nucleus.
  - $[_S$ T is a pointer to the root of a binary tree.$]$ $[_N$ Initialize T.$]$

RST Coherence Relations

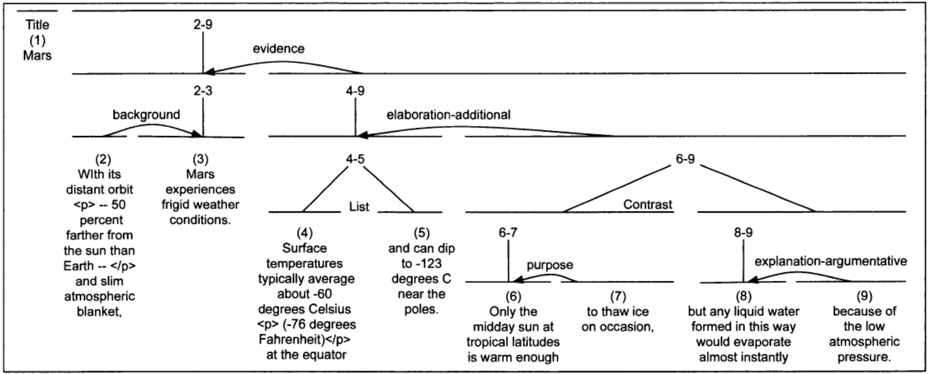

Title
(1)
Mars
2-9
evidence
2-3
background
4-9
elaboration-additional
(2)
With its distant orbit <p> -- 50 percent farther from the sun than Earth -- </p> and slim atmospheric blanket,
(3)
Mars experiences frigid weather conditions.
4-5
List
6-9
Contrast
(4)
Surface temperatures typically average about -60 degrees Celsius <p> (-76 degrees Fahrenheit)</p> at the equator
(5)
and can dip to -123 degrees C near the poles.
6-7
purpose
8-9
explanation-argumentative
(6)
Only the midday sun at tropical latitudes is warm enough
(7)
to thaw ice on occasion,
(8)
but any liquid water formed in this way would evaporate almost instantly
(9)
because of the low atmospheric pressure.

Figure 21.4 A discourse tree for the *Scientific American* text in (21.23), from Marcu (2000a). Note that asymmetric relations are represented with a curved arrow from the satellite to the nucleus.

Automatic Coherence Assignment

- Given a sequence of sentences or clauses, we want to automatically:
  - determine coherence relations between them (coherence relation assignment)
  - extract a tree or graph representing an entire discourse (discourse parsing)

Automatic Coherence Assignment

- **Very difficult!**
- One existing approach is to use cue phrases.
  - John hid Bill's car keys because he was drunk.
  - The scarecrow came to ask for a brain. Similarly, the tin man wants a heart.

1. Identify cue phrases in the text.
2. Segment the text into discourse segments.
   - Use cue phrases/discourse markers
     - although, but, because, yet, with, and, . . .
     - but often implicit, as in car key example
3. Classify the relationship between each consecutive discourse segment.

Reference Resolution

- To interpret the sentence in any discourse we need to who or what entity is being talked about.
- Victoria Chen, CFO of Megabucks Banking Corp since 2004, saw her pay jump 20%, to $1.3 million, as the 37-year-old also became the Denver-based company's president. It has been ten years since she came to Megabucks from rival Lotsaloot.
- Coreference chains:
  - {Victoria Chen, CFO of Megabucks Banking Corp since 2004, her, the 37-year-old, the Denver-based company's president, she}
  - {Megabucks Banking Corp, the Denver-based company, Megabucks}
  - {her pay}
  - {Lotsaloot}

Some Terminology

Victoria Chen, CFO of Megabucks Banking Corp since 2004, saw her pay jump 20%, to $1.3 million, as the 37-year-old also became the Denver-based company's president. It has been ten years since she came to Megabucks from rival Lotsaloot.

- **Referring expression**
  - *Victoria Chen*, *the 37-year-old* and *she* are referring expressions.
- **Referent**
  - **Victoria Chen** is the referent.
- Two referring expressions referring to the same entity are said to **corefer**.
- A referring expression *licenses* the use of a subsequent expression.
  - *Victoria Chen* allows **Victoria Chen** to be referred to as *she*.
  - *Victoria Chen* is the **antecedent** of *she*.
- Reference to an earlier introduced entity is called **anaphora**.
- Such a reference is called **anaphoric**.
  - *the 37-year-old*, *her* and *she* are anaphoric.

References and Context

- Suppose your friend has a car, a 1961 Ford Falcon.
- You can refer to it in many ways: *it, this, that, this car, the car, the Ford, the Falcon, my friend's car*, . . .
- However you are not free to use any of these in any context!
  - For example, you can not refer to it as *it*, or as *the Falcon*, if the hearer has no prior knowledge of the car, or it has not been mentioned, etc.
- Coreference chains are part of cohesion.

Other Kinds of Referents

- You do not always refer to entities. Consider:
- *According to Doug, Sue just bought the Ford Falcon*.
  - *But **that** turned out to be a lie.*
  - *But **that** was false*.
  - ***That** struck me as a funny way to describe the situation*.
  - ***That** caused a financial problem for Sue*.

Types of Referring Expressions

- Indefinite Noun Phrases
- Definite Noun Phrases
- Pronouns
- Demonstratives
- Names

Indefinite Noun Phrases

- Introduce new entities to the discourse
- Usually with *a*, *an*, *some*, and even *this*.
  - Mrs. Martin was so kind as to send Mrs. Goddard *a beautiful goose*.
  - I brought him *some good news*.
  - I saw *this beautiful Ford Falcon* today.
- Specific vs. non-specific ambiguity.
  - The *goose* above is *specific* – it is the one Mrs. Martin sent.
  - The goose in “I am going to the butcher to buy a goose.” is non-specific.

Definite Noun Phrases

- Refer to entities identifiable to the hearer.
- Entities are either previously mentioned:
    - It concerns a while stallion which I have sold to another officer. But the pedigree of *the while stallion* was not fully established.
- Or, they are part of the hearer's beliefs about the world.
    - I read it in *the New York Times*

Pronouns

- Pronouns usually refer to entities that were introduced no further that one or two sentences back.
  - John went to Bob's party and parked next to a classic Ford Falcon.
  - He went inside and talked to Bob for more than a hour. (He = John)
  - Bob told him that he recently got engaged. (him = John, he = Bob)
  - He also said he bought *it* yesterday (He = Bob, it = ???)
  - He also said he bought *the Falcon* yesterday (He = Bob)
- Pronouns can also participate in **cataphora**.
  - Even before *she* saw *it*, Dorothy had been thinking about the statue.
- Pronouns also appear in *quantified* contexts, bound to the quantifier.
  - Every dancer brought *her* left arm forward.

Demonstratives

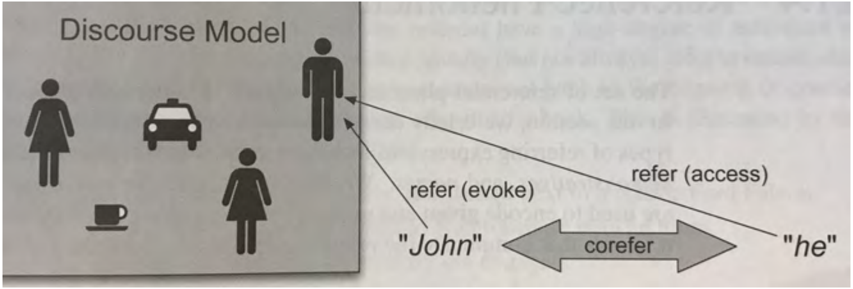
- This, that, these, those
- Can be both pronouns or determiners
  - *That* came earlier.
  - *This* car was parked on the left.
- **Proximal demonstrative** – this
- **Distal demonstrative** – that
- Note that *this NP* is ambiguous: can be both indefinite or definite.

Names

- Names can be used to refer to new or old entities in the discourse.
- These mostly refer to named-entities: people, organizations, locations, geographical objects, products, nationalities, physical facilities, geopolitical entities, dates, monetary instruments, plants, animals, ...
- They are not necessarily unique:
  - *Do you mean the Ali in the sophomore class or the Ali in the senior class?*

Reference Resolution

- Goal: Detemine what entities are referred to by which linguistic expressions.
- The **discourse model** contains our eligible set of referents.



- **Coreference resolution**
- **Pronomial anaphora resolution**

Pronouns Reference Resolution: Filters

- Number, person, gender agreement constraints.
  - *it* can not refer to *books*
  - *she* can not refer to *John*
- Binding theory constraints:
  - John bought himself a new Ford. (himself=John)
  - John bought him a new Ford. (him $\neq$ John)
  - John said that Bill bought him a new Ford. [him $\neq$ Bill]
  - John said that Bill bought himself a new Ford. (himself =Bill)
  - He said that he bought John a new Ford. (both he $\neq$ John)

Pronouns Reference Resolution: Preferences

- **Recency**: preference for most recent referent
- **Grammatical Role**: subj>obj>others
  - Billy went to the bar with Jim. He ordered rum.
- **Repeated mention**:
  - Billy had been drinking for days. He went to the bar again today. Jim went with him. He ordered rum.
- **Parallelism**:
  - John went with Jim to one bar. Bill went with him to another.
- **Verb semantics**:
  - John phoned/criticized Bill. He lost the laptop.
- **Selectional restrictions**:
  - John parked his car in the garage after driving it around for hours.

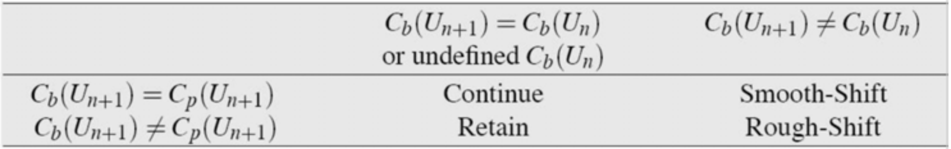
Pronoun Reference Resolution: The Hobbs Algorithm

- Algorithm for walking through parses of current and preceding sentences.
- Simple, often used as baseline.
- Requires parser, morphological gender and number
    - Uses rules to identify heads of NPs
    - Uses WordNet for humanness and gender
        - Is *person* a hypernym of an NP head?
        - Is *female* a hypernym of an NP head?
- Implements binding theory, recency, and grammatical role preferences.

Pronoun Reference Resolution: Centering Theory

- **Claim**: A single entity is “centered” in each sentence.
- That entity is to be distinguished from all other entities that have been evoked.
- Also used in **entity-based coherence**.
- Let U_n and U_{n+1} be two adjacent utterances.
- The **backward-looking center** U_n, denoted $C_b(U_n)$, is the entity focused in the discourse after U_n is interpreted.
- The **forward-looking centers** of U_n, denoted $C_f(U_n)$, is an ordered list of the entities mentioned in U_n of of which could serve as the C_b of U_{n+1}.
- $C_b(U_{n+1})$ in the highest ranked element of $C_f(U_n)$, also mentioned in U_{n+1}.
- Entities in $C_f(U_n))$ are ordered: *subject* $>$ *existential predicate nominal* $>$ object$>$ *indirect object*...
- C_p is the **preferred center**, the first element of $C_f(U_n)$.

Sentence Transitions



	$C_b(U_{n+1}) = C_b(U_n)$ or undefined $C_b(U_n)$	$C_b(U_{n+1}) \neq C_b(U_n)$
$C_b(U_{n+1}) = C_p(U_{n+1})$	Continue	Smooth-Shift
$C_b(U_{n+1}) \neq C_p(U_{n+1})$	Retain	Rough-Shift

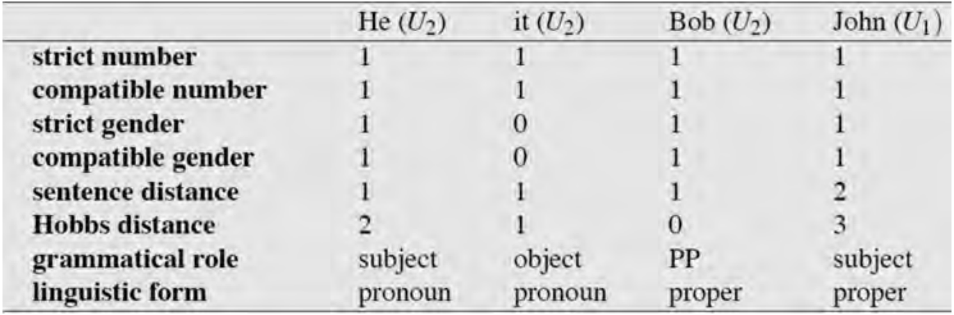
- **Rule 1**: If any element of $C_f(U_n)$ is realized as a pronoun in U_{n+1}, the $C_b(U_{n+1}$ must be realized as a pronoun.
- **Rule 2**:Transition states are ordered: Continue $>$ Retain$>$ Smooth-shift $>$ Rough-shift
- Algorithm:
  - Generate possible C_b– C_f combinations for each possible set of reference assignments.
  - Filter by constraints: agreements, selectional restrictions, centering rules and constraints
  - Rank by transition orderings
  - The most preferred relation defines the pronomial referents.

Pronoun Reference Resolution: Log-Linear Models

- Supervised: hand-labeled coreference corpus
- Rule-based filtering of non-referential pronouns:
  - It was a dark and stormy night.
  - It is raining.
- Needs positive and negative examples:
  - Positive examples in the corpus.
  - Negative examples are created by pairing pronouns with other noun phrases.
- Features are extracted for each training example.
- Classifier learns to predict 1 or 0.
- During testing:
  - Classifier extracts all potential antecedents by parsing the current and previous sentences.
  - Each NP is considered a potential antecedent for each following pronoun.
  - Each pronoun – potential antecedent pair is then presented (through their features) to the classifier.
  - Classifier predicts 1 or 0.

Pronoun Reference Resolution: Log-Linear Models

- Example
  - U_1: John saw a Ford at the dealership.
  - U_2: He showed it to Bob.
  - U_3: He bought it.
- Features for *He* in U_3



	He (U_2)	it (U_2)	Bob (U_2)	John (U_1)
strict number	1	1	1	1
compatible number	1	1	1	1
strict gender	1	0	1	1
compatible gender	1	0	1	1
sentence distance	1	1	1	2
Hobbs distance	2	1	0	3
grammatical role	subject	object	PP	subject
linguistic form	pronoun	pronoun	proper	proper

General Reference Resolution

- Victoria Chen, CFO of Megabucks Banking Corp since 2004, saw her pay jump 20%, to \$1.3 million, as the 37-year-old also became the Denver-based company's president. It has been ten years since she came to Megabucks from rival Lotsaloot.
- Coreference chains:
  - {Victoria Chen, CFO of Megabucks Banking Corp since 2004, her, the 37-year-old, the Denver-based company's president, she}
  - {Megabucks Banking Corp, the Denver-based company, Megabucks}
  - {her pay}
  - {Lotsaloot}

High-level Recipe for Coreference Resolution

- Parse the text and identify NPs; then
- For every pair of NPs, carry out binary classification: coreferential or not?
- Collect the results into coreference chains
- What do we need?
  - A choice of classifier.
  - Lots of labeled data.
  - Features

High-level Recipe for Coreference Resolution

- Word-level edit distance between the two NPs
- Are the two NPs the same NER type?
- Appositive syntax
  - “Alan Shepherd, the first American astronaut, . . . ”
- Proper/definite/indefinite/pronoun
- Gender
- Number
- Distance in sentences
- Number of NPs between
- Grammatical roles
- Any other relevant features,.e.g embeddings?

Pragmatics

- Pragmatics is a branch of linguistics dealing with language use in context.
  - When a diplomat says yes, he means ‘perhaps’;
  - When he says perhaps, he means ‘no’;
  - When he says no, he is not a diplomat.
  - (Variously attributed to Voltaire, H. L. Mencken, and Carl Jung)

In Context?

- Social context
    - Social identities, relationships, and setting
- Physical context
    - Where? What objects are present? What actions?
- Linguistic context
    - Conversation history
- Other forms of context
    - Shared knowledge, etc.

Language as Action: Speech Acts

- The Mood of a sentence indicates relation between speaker and the concept (proposition) defined by the LF
- There can be operators that represent these direct relations:
  - ASSERT: the proposition is proposed as a fact
  - YN-QUERY: the truth of the proposition is queried
  - COMMAND: the proposition describes a requested action
  - WH-QUERY: the proposition describes an object to be identified
- There are also indirect speech acts.
  - Can you pass the salt?
  - It is warm here.

”How to do things with words.” Jane Austin[1]

- In addition to just saying things, *sentences perform actions.*
- When these sentences are uttered, the important thing is not their truth value, but the *felicitousness* of the action (e.g., do you have the *authority* to do it):
  - I name this ship the Titanic.
  - I take this man to be my husband.
  - I bequeath this watch to my brother.
  - I declare war.

[1] http://en.wikipedia.org/wiki/J._L._Austin

Performative Sentences

- When uttered by the proper authority, such sentences have the effect of changing the state of the world, just as any other action that can change the state of the world.
  - These involve verbs like, *name*, *second*, *declare*, etc.
- “I name this ship the Titanic.” also causes the ship to be named *Titanic*.
- You can tell whether sentences are performative by adding “hereby”:
  - I hereby name this ship the Queen Elizabeth.
- Non-performative sentences do not sound good with hereby:
  - Birds hereby sing.
  - There is hereby fighting in Syria.

Speech Acts Continued

- **Locutionary Act**: The utterance of a sentence with a particular meaning.
- **Illocutionary Act**: The act of asking, answering, promising, etc. in uttering a sentence.
  - I promise you that I will fix the problem.
  - You can't do that (protesting)
  - By the way, I have a CD of Debussy; would you like to borrow it? (offering)
- **Perlocutionary Act**: The – often intentional – production of certain effects on the addressee.
  - You can't do that. (stopping or annoying the addressee)
  - By the way, I have a CD of Debussy; would you like to borrow it? (impressing the addressee)

Searle's Speech Acts

- **Assertives** = speech acts that commit a speaker to the truth of the expressed proposition
- **Directives** = speech acts that are to cause the hearer to take a particular action, e.g. requests, commands and advice
  - Can you pass the salt?
  - Has the form of a *question* but the effect of a *directive*
- **Commissives** = speech acts that commit a speaker to some future action, e.g. promises and oaths
- **Expressives** = speech acts that express the speaker's attitudes and emotions towards the proposition, e.g. congratulations, excuses
- **Declarations** = speech acts that change the reality in accord with the proposition of the declaration, e.g. pronouncing someone guilty or pronouncing someone husband and wife

Speech Acts in NLP

- Speech acts (inventories) are mainly used in developing (task-oriented) **dialog systems**.
- Speech acts are used as annotation guidelines for corpus annotation.
- An annotated corpus is then used for machine learning of dialog tasks.
- Such corpora are highly developed and checked for intercoder agreement.
    - Annotation still takes a long time to learn.

Task-oriented Dialogues

- Making travel reservations (flight, hotel room, etc.)
- Scheduling a meeting.
- Finding out when the next bus is.
- Making a payment over the phone.

Ways of Asking for a Room

- I'd like to make a reservation
- I'm calling to make a reservation
- Do you have a vacancy on . . .
- Can I reserve a room?
- Is it possible to reserve a room?

Examples of Task-oriented Speech Acts

- **Identify self**:
  - This is David
  - My name is David
  - I'm David
  - David here
- **Sound check**: Can you hear me?
- **Meta dialogue act**: There is a problem.
- **Greet**: Hello.
- **Request-information**:
  - Where are you going.
  - Tell me where you are going.

Examples of Task-oriented Speech Acts

- **Backchannel** – Sounds you make to indicate that you are still listening
  - ok, m-hm
- **Apologize/reply to apology**
- **Thank/reply to thanks**
- **Request verification/Verify**
  - So that's 2:00? Yes. 2:00.
- **Resume topic**
  - Back to the accommodations . . .
- **Answer a yes/no question**: yes, no.

Task-oriented Speech Acts in Negotiation

- **Suggest**
  - I recommend this hotel
- **Offer**
  - I can send some brochures.
  - How about if I send some brochures.
- **Accept**
  - Sure. That sounds fine.
- **Reject**
  - No. I don't like that one.

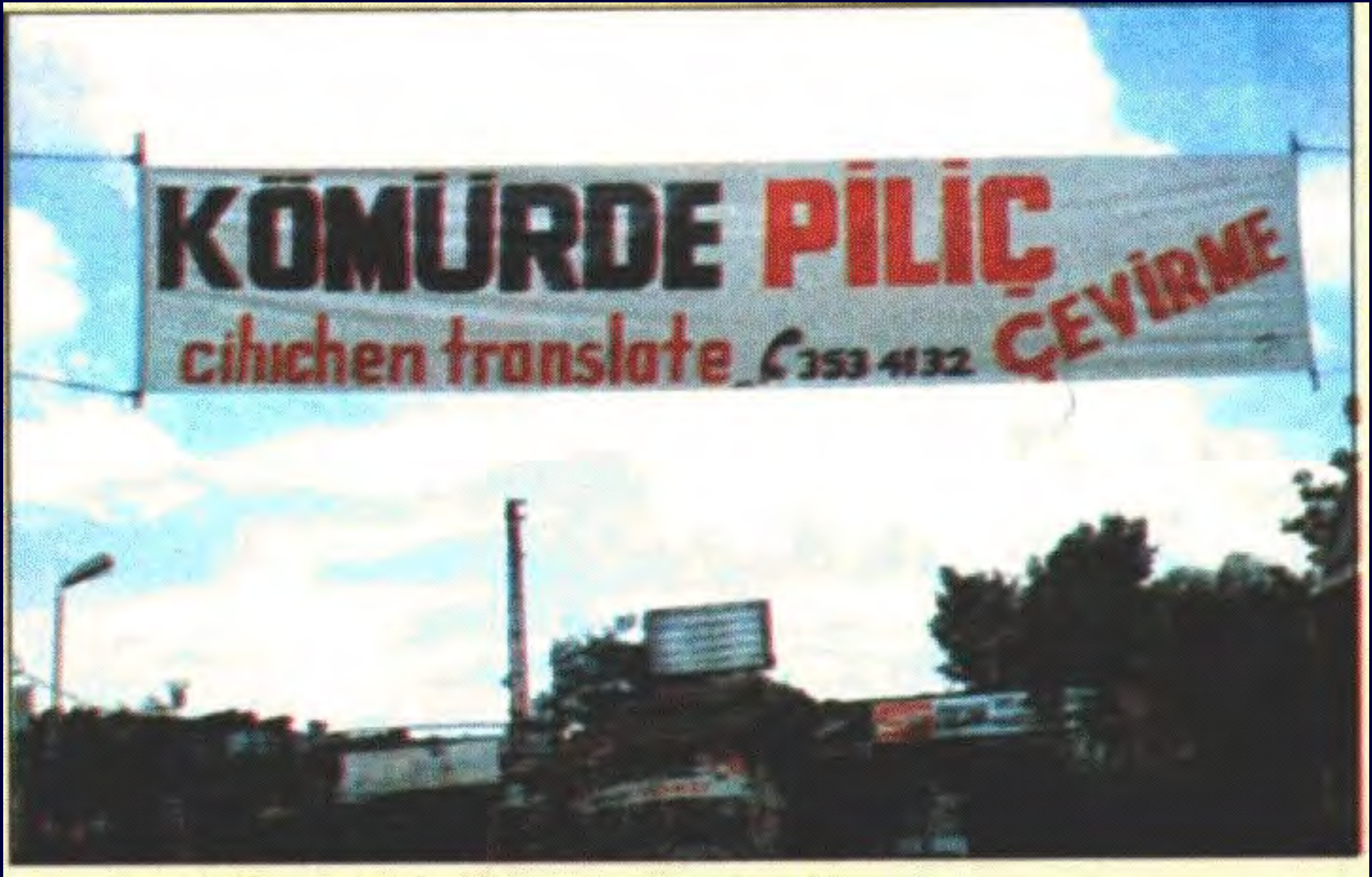
Negotiation

"No, Thursday's out. How about never—is never good for you?"

(Mostly Statistical) Machine Translation

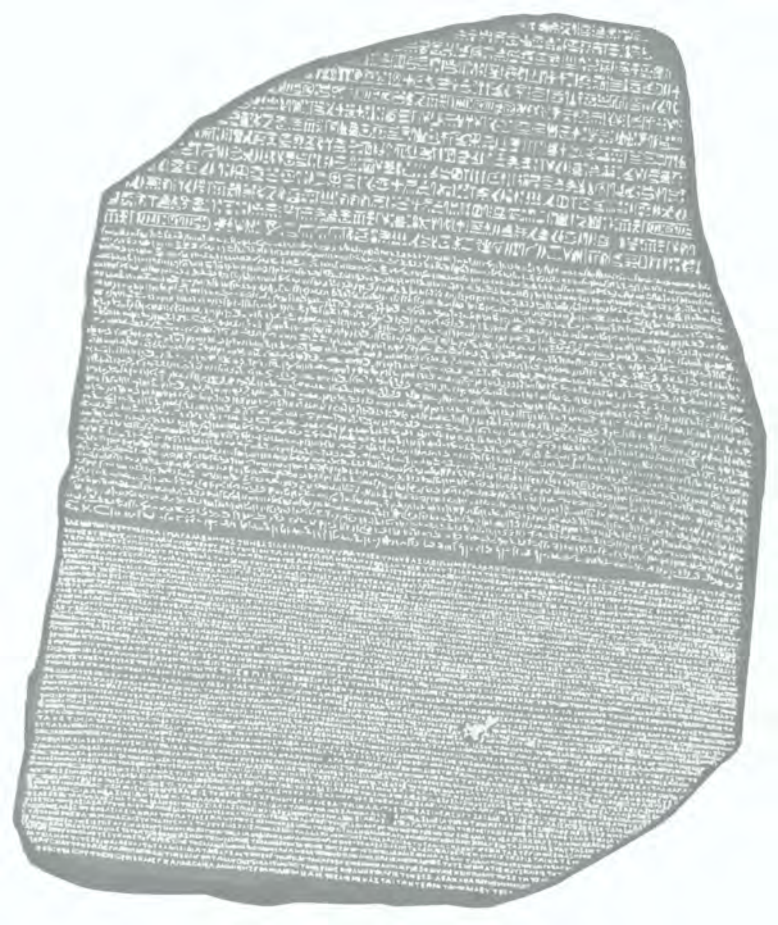
11-411

Fall 2017


KÖMÜRDE PİLİÇ
ÇEVİRME
cihıchen translate
☎353 4132

The Rosetta Stone

- Decree from Ptolemy V on repealing taxes and erecting some statues (196 BC)
- Written in three languages
  - Hieroglyphic
  - Demotic
  - Classical Greek



Overview

- History of Machine Translation
- Early Rule-based Approaches
- Introduction to Statistical Machine Translation (SMT)
- Advanced Topics in SMT
- Evaluation of (S)MT output

Machine Translation

- Transform text (speech) in one language (source) to text (speech) in a different language (target) such that
  - The “meaning” in the source language input is (mostly) preserved, and
  - The target language output is grammatical.
- Holy grail application in AI/NLP since middle of 20th century.

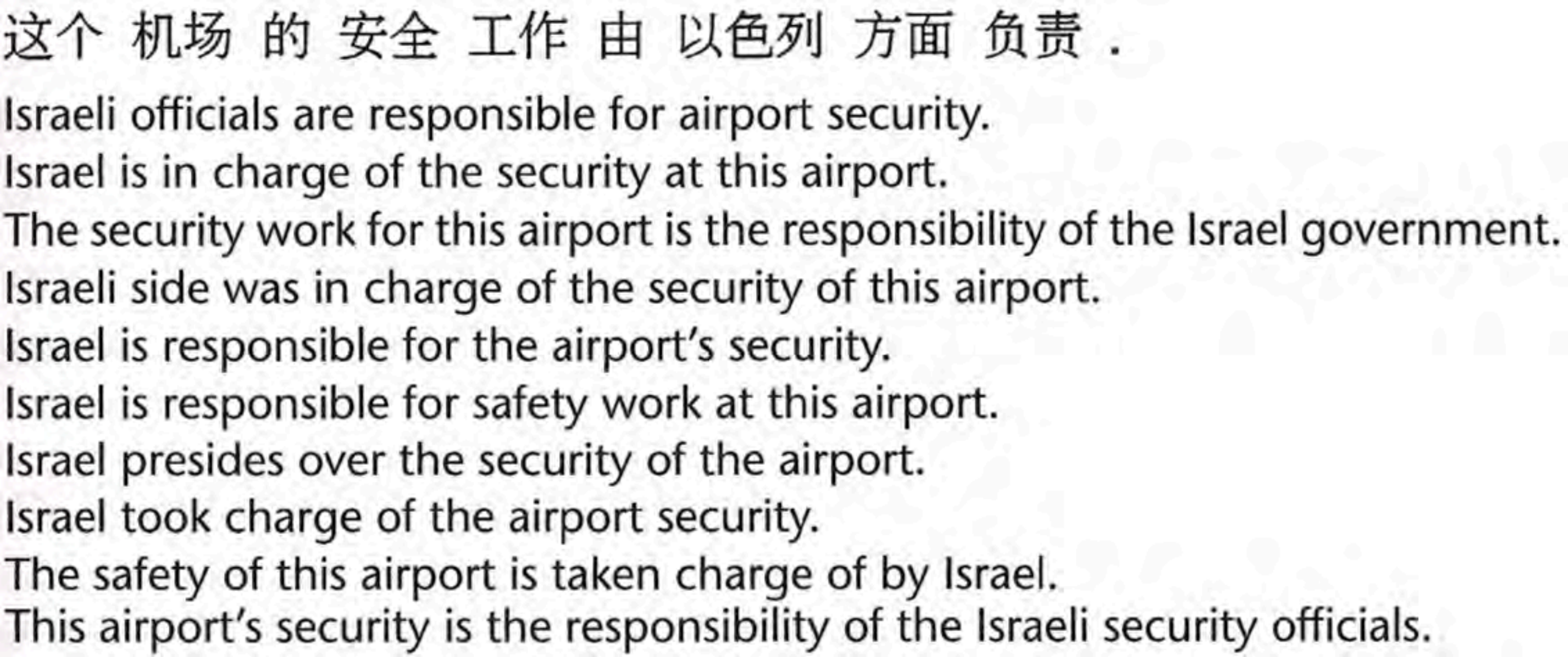
Translation

- Process
  - Read the text in the source language
  - **Understand** it
  - **Write** it down in the target language

- These are hard tasks for computers
  - The human process is invisible, intangible

Machine Translation

Many possible legitimate translations!



这个 机场 的 安全 工作 由 以色列 方面 负责 .

Israeli officials are responsible for airport security.
Israel is in charge of the security at this airport.
The security work for this airport is the responsibility of the Israel government.
Israeli side was in charge of the security of this airport.
Israel is responsible for the airport's security.
Israel is responsible for safety work at this airport.
Israel presides over the security of the airport.
Israel took charge of the airport security.
The safety of this airport is taken charge of by Israel.
This airport's security is the responsibility of the Israeli security officials.

Machine Translation

Rolls-Royce Merlin Engine (from German Wikipedia)

- Der Rolls-Royce Merlin ist ein 12-Zylinder-Flugmotor von Rolls-Royce in V-Bauweise, der vielen wichtigen britischen und US-amerikanischen Flugzeugmustern des ZweitenWeltkriegs als Antrieb diente. Ab 1941 wurde der Motor in Lizenz von der Packard Motor Car Company in den USA als Packard V-1650 gebaut.
- Nach dem Krieg wurden diverse Passagier- und Frachtflugzeuge mit diesem Motor ausgestattet, so z. B. Avro Lancastrian, Avro Tudor und Avro York, später noch einmal die Canadair C-4 (umgebaute Douglas C-54). Der zivile Einsatz des Merlin hielt sich jedoch in Grenzen, da er als robust, aber zu laut galt.
- Die Bezeichnung des Motors ist gemäß damaliger Rolls-Royce Tradition von einer Vogelart, dem Merlinfalken, übernommen und nicht, wie oft vermutet, von dem Zauberer Merlin.

English Translation (via Google Translate)

- The Rolls-Royce Merlin is a 12-cylinder aircraft engine from Rolls-Royce V-type, which served many important British and American aircraft designs of World War II as a drive. From 1941 the engine was built under license by the Packard Motor Car Company in the U.S. as a Packard V-1650th.
- After the war, several passenger and cargo aircraft have been equipped with this engine, such as Avro Lancastrian, Avro Tudor Avro York and, later, the Canadair C-4 (converted Douglas C-54). The civilian use of the Merlin was, however, limited as it remains robust, however, was too loud.
- The name of the motor is taken under the then Rolls-Royce tradition of one species, the Merlin falcon, and not, as often assumed, by the wizard Merlin.

Machine Translation

Rolls-Royce Merlin Engine (from German Wikipedia)

- Der Rolls-Royce Merlin ist ein 12-Zylinder-Flugmotor von Rolls-Royce in V-Bauweise, der vielen wichtigen britischen und US-amerikanischen Flugzeugmustern des Zweitenweltkriegs als Antrieb diente. Ab 1941 wurde der Motor in Lizenz von der Packard Motor Car Company in den USA als Packard V-1650 gebaut.
- Nach dem Krieg wurden diverse Passagier- und Frachtflugzeuge mit diesem Motor ausgestattet, so z. B. Avro Lancastrian, Avro Tudor und Avro York, später noch einmal die Canadair C-4 (umgebaute Douglas C-54). Der zivile Einsatz des Merlin hielt sich jedoch in Grenzen, da er als robust, aber zu laut galt.
- Die Bezeichnung des Motors ist gemäß damaliger Rolls-Royce Tradition von einer Vogelart, dem Merlinfalken, übernommen und nicht, wie oft vermutet, von dem Zauberer Merlin.

English Translation (via Google Translate)

- The Rolls-Royce Merlin is a 12-cylinder aircraft engine from Rolls-Royce V-type, which served many important British and American aircraft designs of World War II as a drive. From 1941 the engine was built under license by the Packard Motor Car Company in the U.S. as a Packard V-1650th.
- After the war, several passenger and cargo aircraft have been equipped with this engine, such as Avro Lancastrian, Avro Tudor Avro York and, later, the Canadair C-4 (converted Douglas C-54). The civilian use of the Merlin was, however, limited as it remains robust, however, was too loud.
- The name of the motor is taken under the then Rolls-Royce tradition of one species, the Merlin falcon, and not, as often assumed, by the wizard Merlin.

Machine Translation

Rolls-Royce Merlin Engine (from German Wikipedia)

- Der Rolls-Royce Merlin ist ein 12-Zylinder-Flugmotor von Rolls-Royce in V-Bauweise, der vielen wichtigen britischen und US-amerikanischen Flugzeugmustern des ZweitenWeltkriegs als Antrieb diente. Ab 1941 wurde der Motor in Lizenz von der Packard Motor Car Company in den USA als Packard V-1650 gebaut.
- Nach dem Krieg wurden diverse Passagier- und Frachtflugzeuge mit diesem Motor ausgestattet, so z. B. Avro Lancastrian, Avro Tudor und Avro York, später noch einmal die Canadair C-4 (umgebaute Douglas C-54). Der zivile Einsatz des Merlin hielt sich jedoch in Grenzen, da er als robust, aber zu laut galt.
- Die Bezeichnung des Motors ist gemäß damaliger Rolls-Royce Tradition von einer Vogelart, dem Merlinfalken, übernommen und nicht, wie oft vermutet, von dem Zauberer Merlin.

Turkish Translation (via Google Translate)

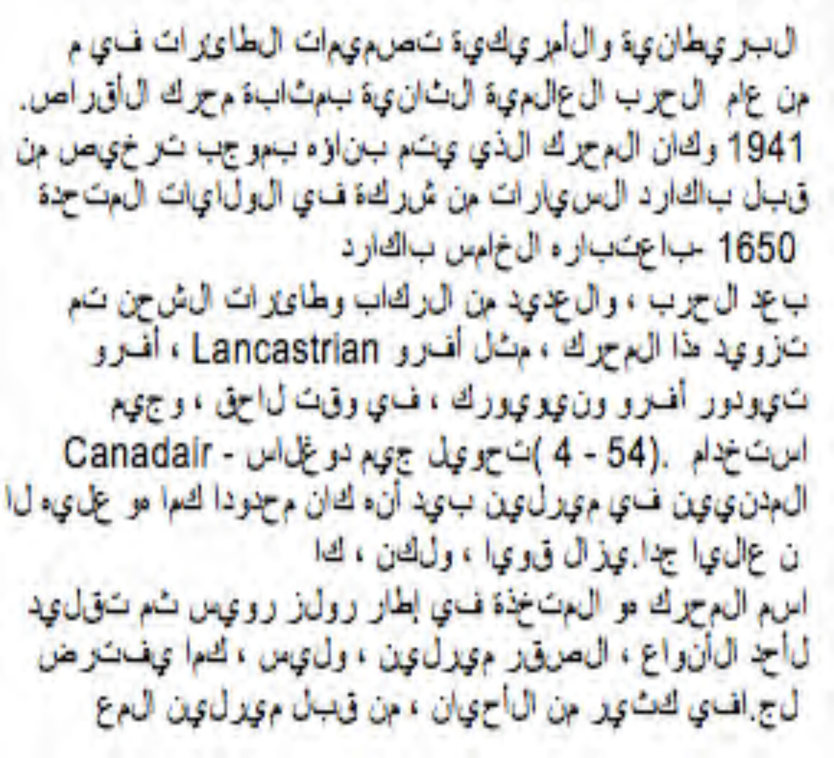
- Rolls-Royce Merlin 12-den silindirli Rolls-Royce uçak motoru V tipi, bir sürücü olarak Dünya Savaşı'nın birçok önemli İngiliz ve Amerikan uçak tasarımları devam eder. 1.941 motor lisansı altında Packard Motor Car Company tarafından ABD'de Packard V olarak yaptırılmıştır Gönderen-1650
- Savaştan sonra, birkaç yolcu ve kargo uçakları ile Avro Lancastrian, Avro Avro York ve Tudor gibi bu motor, daha sonra, Canadair C-4 (Douglas C-54) dönüştürülür donatılmıştır. Olarak, ancak, çok yüksek oldu sağlam kalır Merlin sivil kullanıma Ancak sınırlıydı.
- Motor adı daha sonra Rolls altında bir türün, Merlin şahin, ve değil-Royce geleneği, sıklıkta kabul, Merlin sihirbaz tarafından alınır.

Machine Translation

Rolls-Royce Merlin Engine (from German Wikipedia)

- Der Rolls-Royce Merlin ist ein 12-Zylinder-Flugmotor von Rolls-Royce in V-Bauweise, der vielen wichtigen britischen und US-amerikanischen Flugzeugmustern des ZweitenWeltkriegs als Antrieb diente. Ab 1941 wurde der Motor in Lizenz von der Packard Motor Car Company in den USA als Packard V-1650 gebaut.
- Nach dem Krieg wurden diverse Passagier- und Frachtflugzeuge mit diesem Motor ausgestattet, so z. B. Avro Lancastrian, Avro Tudor und Avro York, später noch einmal die Canadair C-4 (umgebaute Douglas C-54). Der zivile Einsatz des Merlin hielt sich jedoch in Grenzen, da er als robust, aber zu laut galt.
- Die Bezeichnung des Motors ist gemäß damaliger Rolls-Royce Tradition von einer Vogelart, dem Merlinfalken, übernommen und nicht, wie oft vermutet, von dem Zauberer Merlin.

Arabic Translation (via Google Translate -- 2009



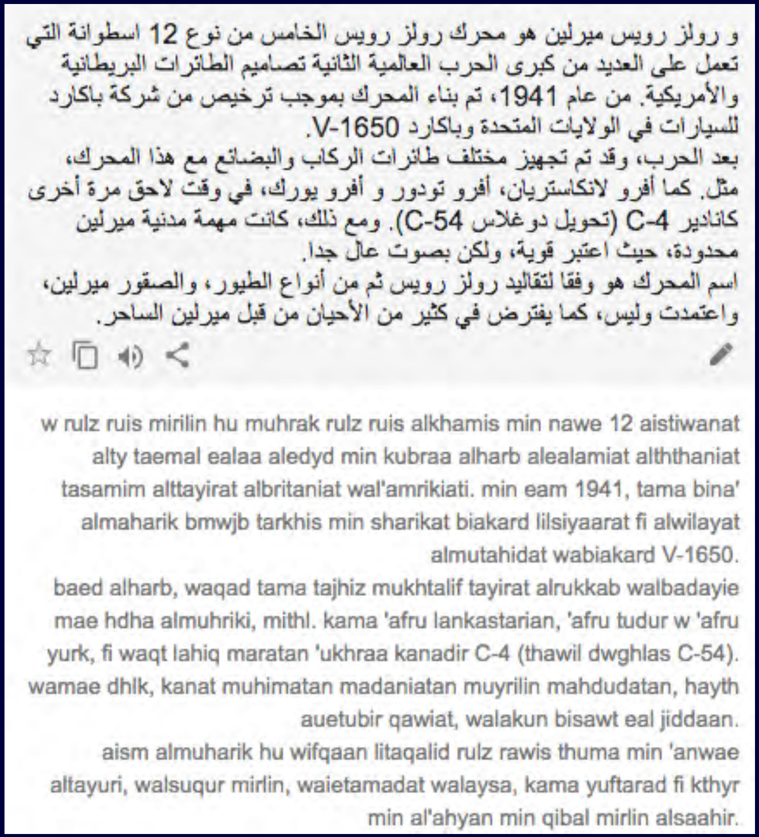
البريطانية والأمريكية تصميمات الطائرات في م
من عام الحرب العالمية الثانية بمثابة محرك الأقراص.
1941 وكان المحرك الذي يتم بناؤه بموجب ترخيص من
قبل باكارد السيارات من شركة في الولايات المتحدة
1650 ـباعتباره الخامس باكارد
بعد الحرب ، والعديد من الركاب وطائرات الشحن تم
تزويد هذا المحرك ، مثل أفرو Lancastrian ، أفرو
تيودور أفرو ونيويورك ، في وقت لاحق ، وجيم
استخدام .(4 - 54)تحويل جيم دوغلاس - Canadair
المدنيين في ميرلين بيد أنه كان محدودا كما هو عليه ال
ن عاليا جدا.يزال قويا ، ولكن ، كا
اسم المحرك هو المتخذة في إطار رولز رويس ثم تقليد
لأحد الأنواع ، الصقر ميرلين ، ولويس ، كما يفترض
لج.افي كثير من الأحيان ، من قبل ميرلين المع

Machine Translation

Rolls-Royce Merlin Engine (from German Wikipedia)

- Der Rolls-Royce Merlin ist ein 12-Zylinder-Flugmotor von Rolls-Royce in V-Bauweise, der vielen wichtigen britischen und US-amerikanischen Flugzeugmustern des ZweitenWeltkriegs als Antrieb diente. Ab 1941 wurde der Motor in Lizenz von der Packard Motor Car Company in den USA als Packard V-1650 gebaut.
- Nach dem Krieg wurden diverse Passagier- und Frachtflugzeuge mit diesem Motor ausgestattet, so z. B. Avro Lancastrian, Avro Tudor und Avro York, später noch einmal die Canadair C-4 (umgebaute Douglas C-54). Der zivile Einsatz des Merlin hielt sich jedoch in Grenzen, da er als robust, aber zu laut galt.
- Die Bezeichnung des Motors ist gemäß damaliger Rolls-Royce Tradition von einer Vogelart, dem Merlinfalken, übernommen und nicht, wie oft vermutet, von dem Zauberer Merlin.

Arabic Translation (via Google Translate – 2017)



و رولز رويس ميرلين هو محرك رولز رويس الخامس من نوع 12 اسطوانة التي تعمل على العديد من كبرى الحرب العالمية الثانية تصاميم الطائرات البريطانية والأمريكية. من عام 1941، تم بناء المحرك بموجب ترخيص من شركة باكارد للسيارات في الولايات المتحدة وباكارد V-1650.

بعد الحرب، وقد تم تجهيز مختلف طائرات الركاب والبضائع مع هذا المحرك، مثل. كما أفرو لانكاستريان، أفرو تودور و أفرو يورك، في وقت لاحق مرة أخرى كانادير C-4 (تحويل دوغلاس C-54). ومع ذلك، كانت مهمة مدنية ميرلين محدودة، حيث اعتبر قوية، ولكن بصوت عال جدا.

اسم المحرك هو وفقا لتقاليد رولز رويس ثم من أنواع الطيور ، والصقور ميرلين، واعتمدت وليس، كما يفترض في كثير من الأحيان من قبل ميرلين الساحر.

w rulz ruis mirilin hu muhrak rulz ruis alkhamis min nawe 12 aistiwanat alty taemal ealaa aledyd min kubraa alharb alealamiat alththaniat tasamim alttayirat albritaniat wal'amrikiati. min eam 1941, tama bina' almaharik bmwjb tarkhis min sharikat biakard lilsiyaarat fi alwilayat almutahidat wabiakard V-1650.

baed alharb, waqad tama tajhiz mukhtalif tayirat alrukkab walbadayie mae hdha almuhriki, mithl. kama 'afru lankastarian, 'afru tudur w 'afru yurk, fi waqt lahiq maratan 'ukhraa kanadir C-4 (thawil dwghlas C-54). wamae dhlk, kanat muhimatan madaniatan muyrilin mahdudatan, hayth auetubir qawiat, walakun bisawt eal jiddaan.

aism almuharik hu wifqaan litaqalid rulz rawis thuma min 'anwae altayuri, walsuqur mirlin, waietamadat walaysa, kama yuftarad fi kthyr min al'ahyan min qibal mirlin alsaahir.

Machine Translation

- (Real-time speech-to-speech) Translation is a very demanding task
  - – Simultaneous translators (in UN, or EU Parliament) last about 30 minutes
  - – Time pressure
  - – Divergences between languages
    - German: Subject ……………………… Verb
    - English: Subject Verb ………………………
    - Arabic: Verb Subject ………….

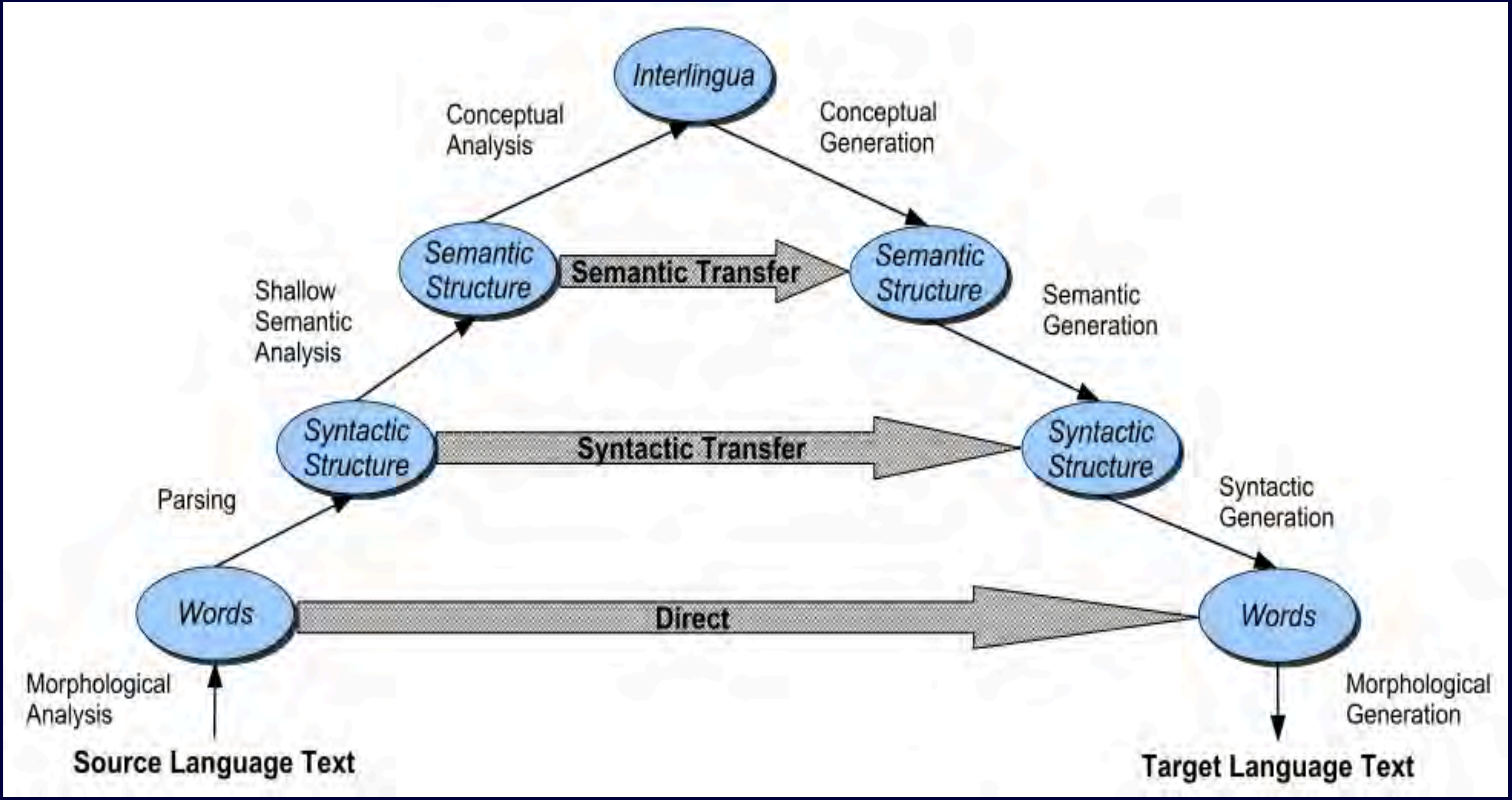
Brief History

- 1950’s: Intensive research activity in MT
  - Translate Russian into English
- 1960’s: Direct word-for-word replacement
- 1966 (ALPAC): NRC Report on MT
  - Conclusion: MT no longer worthy of serious scientific investigation.
- 1966-1975: `Recovery period’
- 1975-1985: Resurgence (Europe, Japan)
- 1985-present: Resurgence (US)
  - Mostly Statistical Machine Translation since 1990s
  - Recently Neural Network/Deep Learning based machine translation

Early Rule-based Approaches

- Expert system-like rewrite systems
- Interlingua methods (analyze and generate)
- Information used for translation are compiled by humans
  - Dictionaries
  - Rules

Vauquois Triangle



Statistical Approaches

- Word-to-word translation
- Phrase-based translation
- Syntax-based translation (tree-to-tree, tree-to-string)
  - Trained on parallel corpora
  - Mostly noisy-channel (at least in spirit)

Early Hints on the Noisy Channel Intuition

- “One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: ‘This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.’ ”

Warren Weaver

  - (1955:18, quoting a letter he wrote in 1947)

Divergences between Languages

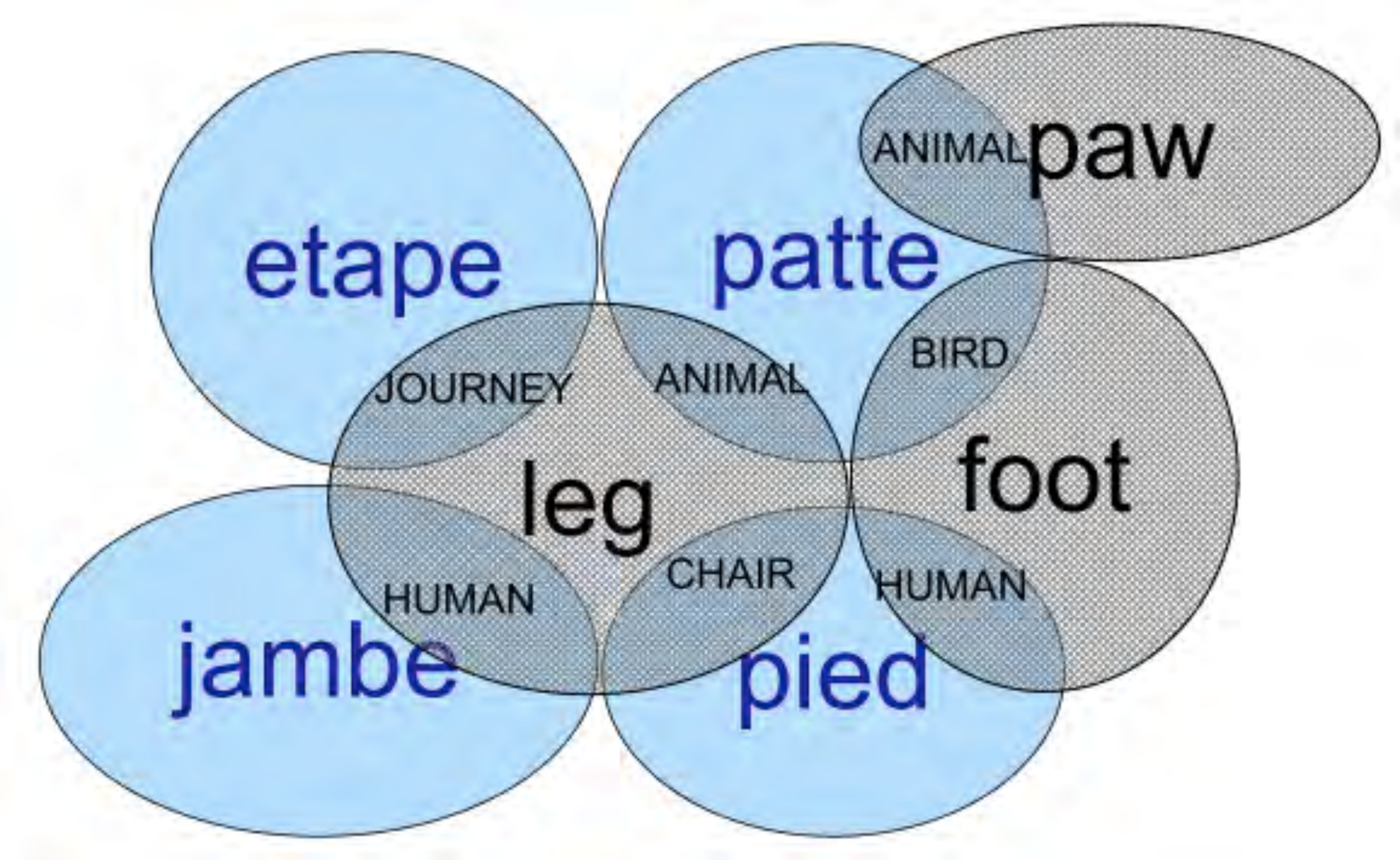
- Languages differ along many dimensions
  - Concept – Lexicon alignment – Lexical Divergence
  - Syntax – Structure Divergence
    - Word-order differences
      - English is Subject-Verb-Object
      - Arabic is Verb-Subject-Object
      - Turkish is Subject-Object-Verb
    - Phrase order differences
    - Structure-Semantics Divergences

Lexical Divergences

- English: wall
  - German: Wand for walls inside, Mauer for walls outside
- English: runway
  - Dutch: Landingbaan for when you are landing; startbaan for when you are taking off
- English: aunt
  - Turkish: hala (father's sister), teyze(mother's sister)
- Turkish: o
  - English: she, he, it

Lexical Divergences

How conceptual space is cut up



Lexical Gaps

- One language may not have a word for a concept in another language
  - Japanese: oyakoko
    - Best English approximation: “filial piety”
  - Turkish: gurbet
    - Where you are when you are not “home”
  - English: condiments
    - Turkish: ??? (things like mustard, mayo and ketchup)

Local Phrasal Structure Divergences

- English: a blue house
  - French: une maison bleu
- German: die ins Haus gehende Frau
  - English: the lady walking into the house

Structural Divergences

- English: I have a book.
  - Turkish: Benim kitabim var. (Lit: My book exists)
- French: Je m’appelle Jean (Lit: I call myself Jean)
  - English: My name is Jean.
- English: I like swimming.
  - German: Ich schwimme gerne. (Lit: I swim “likingly”.)

Major Rule-based MT Systems/Projects

- Systran
  - Major human effort to construct large translation dictionaires + limited word-reordering rules
- Eurotra
  - Major EU-funded project (1970s-1994) to translate among (then) 12 EC languages.
    - Bold technological framework
      - Structural Interlingua
    - Management failure
    - Never delivered a working MT system
    - Helped create critical mass of researchers

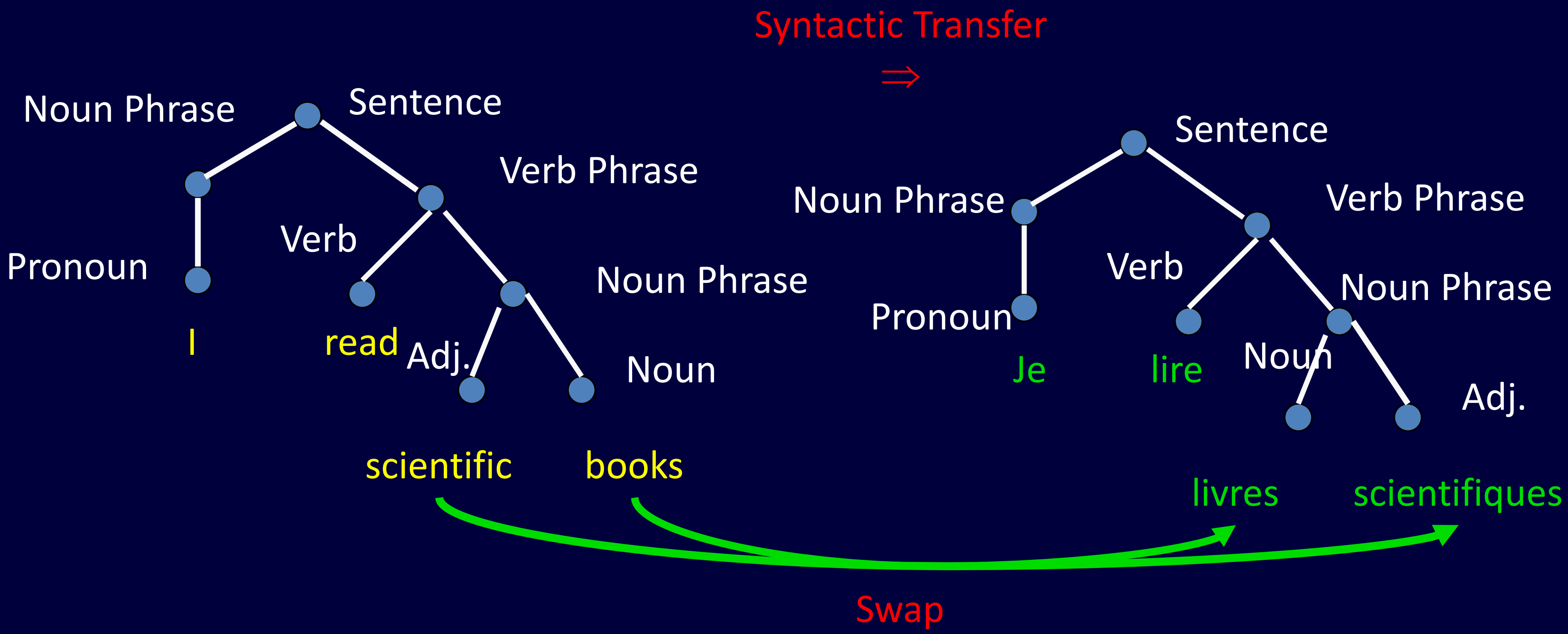
Major Rule-based MT Systems/Projects

- METEO
  - Successful system for French-English translation of Canadian weather reports (1975-1977)
- PANGLOSS
  - Large-scale MT project by CMU/USC-ISI/NMSU
  - Interlingua-based Japanese-Spanish-English translation
  - Manually developed semantic lexicons

Rule-based MT

- Manually develop rules to analyze the source language sentence (e.g., a parser)
  - => some source structure representation
- Map source structure to a target structure
- Generate target sentence from the transferred structure

Rule-based MT



Source language analysis

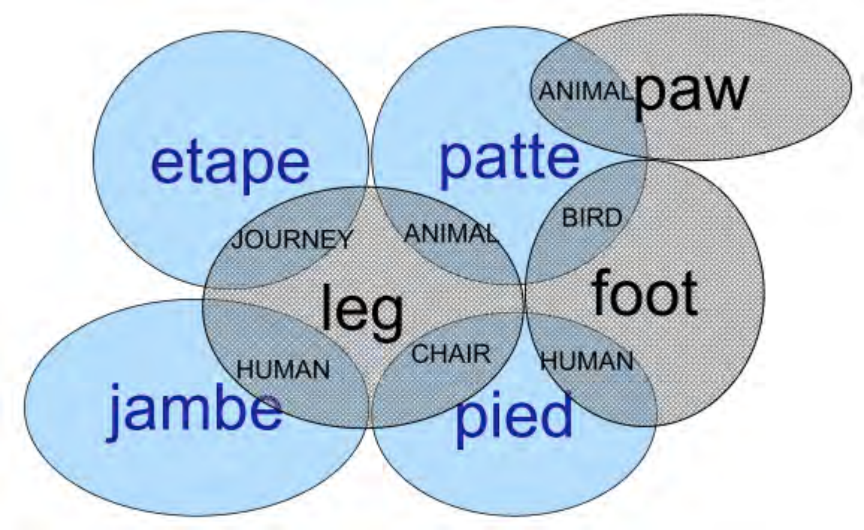
Target language generation

Rules

- Rules to analyze the source sentences
  - (Usually) Context-free grammar rules coupled with linguistic features
    - Sentence => Subject-NP Verb-Phrase
    - Verb-Phrase => Verb Object …..

Rules

- Lexical transfer rules
  - English: book (N) => French: livre (N, masculine)
  - English: pound (N, monetary sense)=> French: livre (N, feminine)
  - English: book (V) => French: réserver (V)
- Quite tricky for



Rules

- Structure Transfer Rules
  - English: S => NP VP ➔

    French: TR(S) => TR(NP) TR(VP)
  - English: NP => Adj Noun ➔

    French: TR(NP) => Tr(Noun) Tr(Adj)

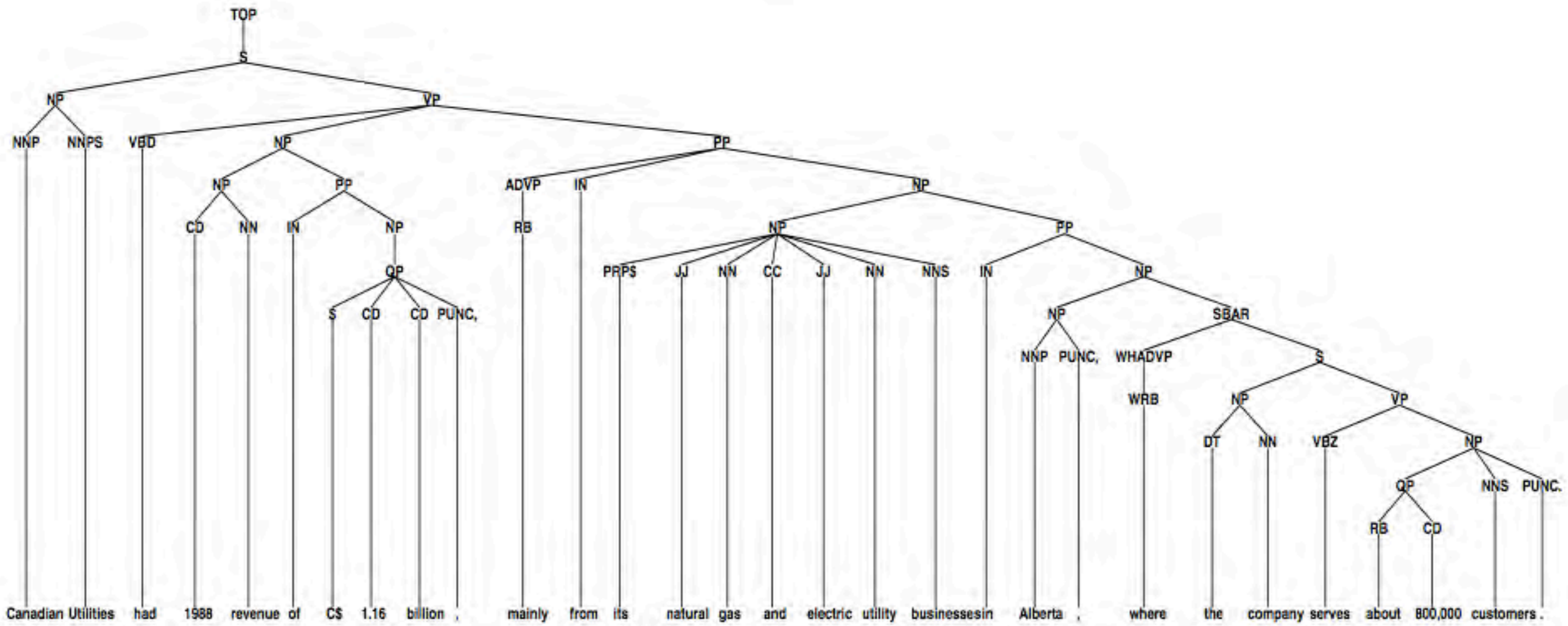
    but there are exceptions for

    Adj=grand, petit, ….

Rules

Much more complex to deal with “real world” sentences.

Canadian Utilities had 1988 revenue of C$ 1.16 billion , mainly from its natural gas and electric utility businesses in Alberta , where the company serves about 800,000 customers .


TOP
S
NP VP
NNP NNPS VBD NP PP
NP PP ADVP IN NP
CD NN IN NP RB NP PP
QP PRP$ JJ NN CC JJ NN NNS IN NP
$ CD CD PUNC, NP SBAR
NNP PUNC, WHADVP S
WRB NP VP
DT NN VBZ NP
QP NNS PUNC.
RB CD
Canadian Utilities had 1988 revenue of C$ 1.16 billion , mainly from its natural gas and electric utility businessesin Alberta , where the company serves about 800,000 customers .

Example-based MT (EBMT)

- Characterized by its use of a bilingual corpus with parallel texts as its main knowledge base, at run-time.
- Essentially translation by analogy and can be viewed as an implementation of case-based reasoning approach of machine learning.
- Find how (parts of) input are translated in the examples
  - Cut and paste to generate novel translations

Example-based MT (EBMT)

- Translation Memory
  - Store many translations,
    - source – target sentence pairs
  - For new sentences, find closes match
    - use edit distance, POS match, other similarity techniques
  - Do corrections,
    - map insertions, deletions, substitutions onto target sentence
  - Useful only when you expect same or similar sentence to show up again, but then high quality

Example-based MT (EBMT)

English	Japanese
• How much is that red umbrella?	• Ano akai kasa wa ikura desu ka?
• How much is that small camera?	• Ano chiisai kamera wa ikura desu ka?
• How much is that X?	• Ano X wa ikura desu ka?

Hybrid Machine Translation

- Use multiple techniques (rule-based/ EBMT/Interlingua)
- Combine the outputs of different systems to improve final translations

How do we evaluate MT output?

- **Adequacy**: Is the meaning of the source sentence conveyed by the target sentence?
- **Fluency**: Is the sentence grammatical in the target language?
- These are rated on a scale of 1 to 5

How do we evaluate MT output?

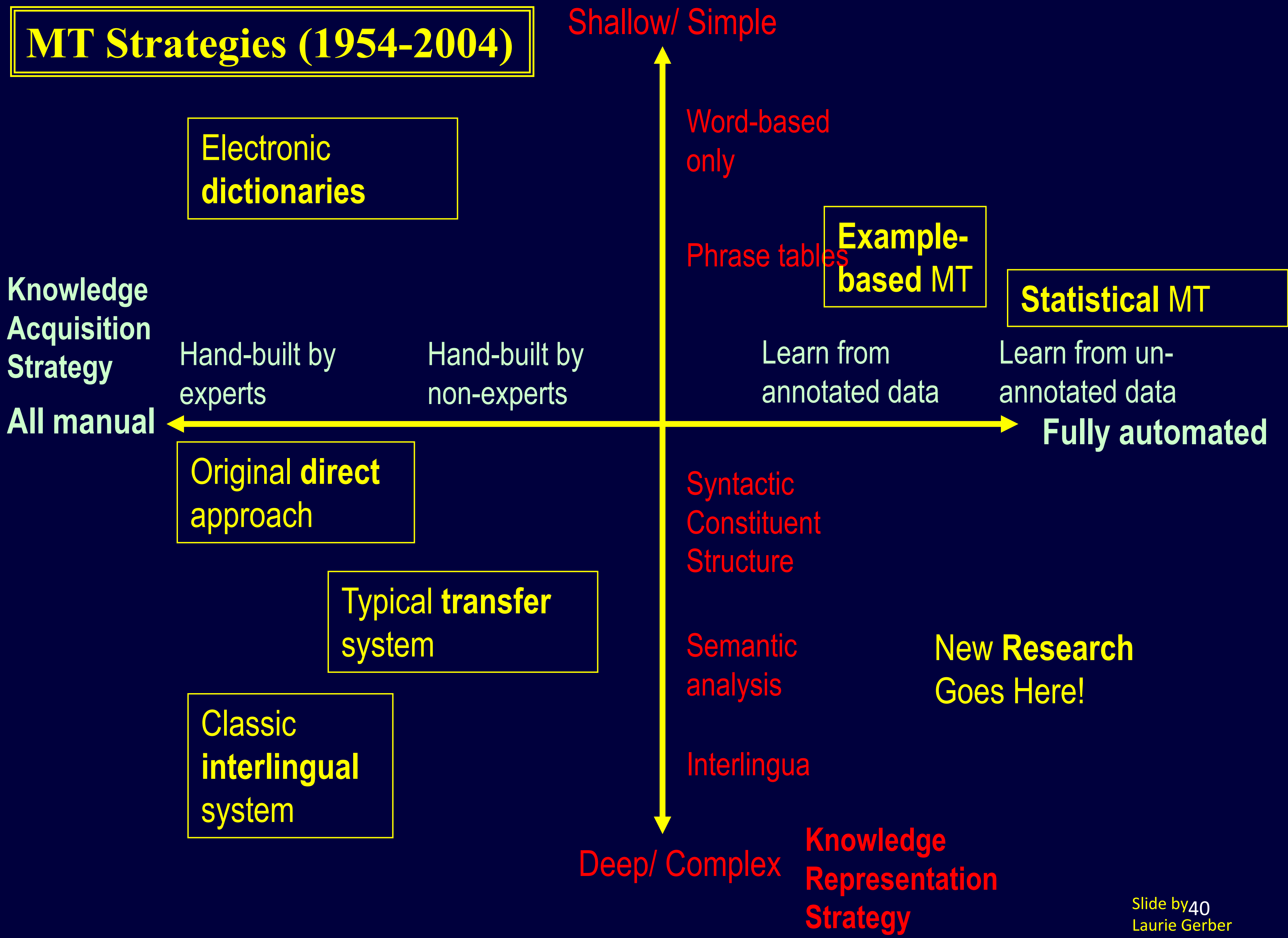
Je suis fatigué.

	Adequacy	Fluency
Tired is I.	5	2
Cookies taste good!	1	5
I am tired.	5	5

How do we evaluate MT output?

- This in general is very labor intensive
  - Read each source sentence
  - Evaluate target sentence for adequacy and fluency
- Not easy to do if you improve your MT system 10 times a day, and need to evaluate!
  - Could this be mechanized?
    - Later

MT Strategies (1954-2004)



Shallow/ Simple

Word-based only

Phrase tables

Syntactic Constituent Structure

Semantic analysis

Interlingua

Deep/ Complex

Knowledge Representation Strategy

Knowledge Acquisition Strategy

All manual

Hand-built by experts

Hand-built by non-experts

Learn from annotated data

Learn from un-annotated data

Fully automated

Electronic **dictionaries**

Original **direct** approach

Typical **transfer** system

Classic **interlingual** system

Example-based MT

Statistical MT

New **Research** Goes Here!

Slide by Laurie Gerber

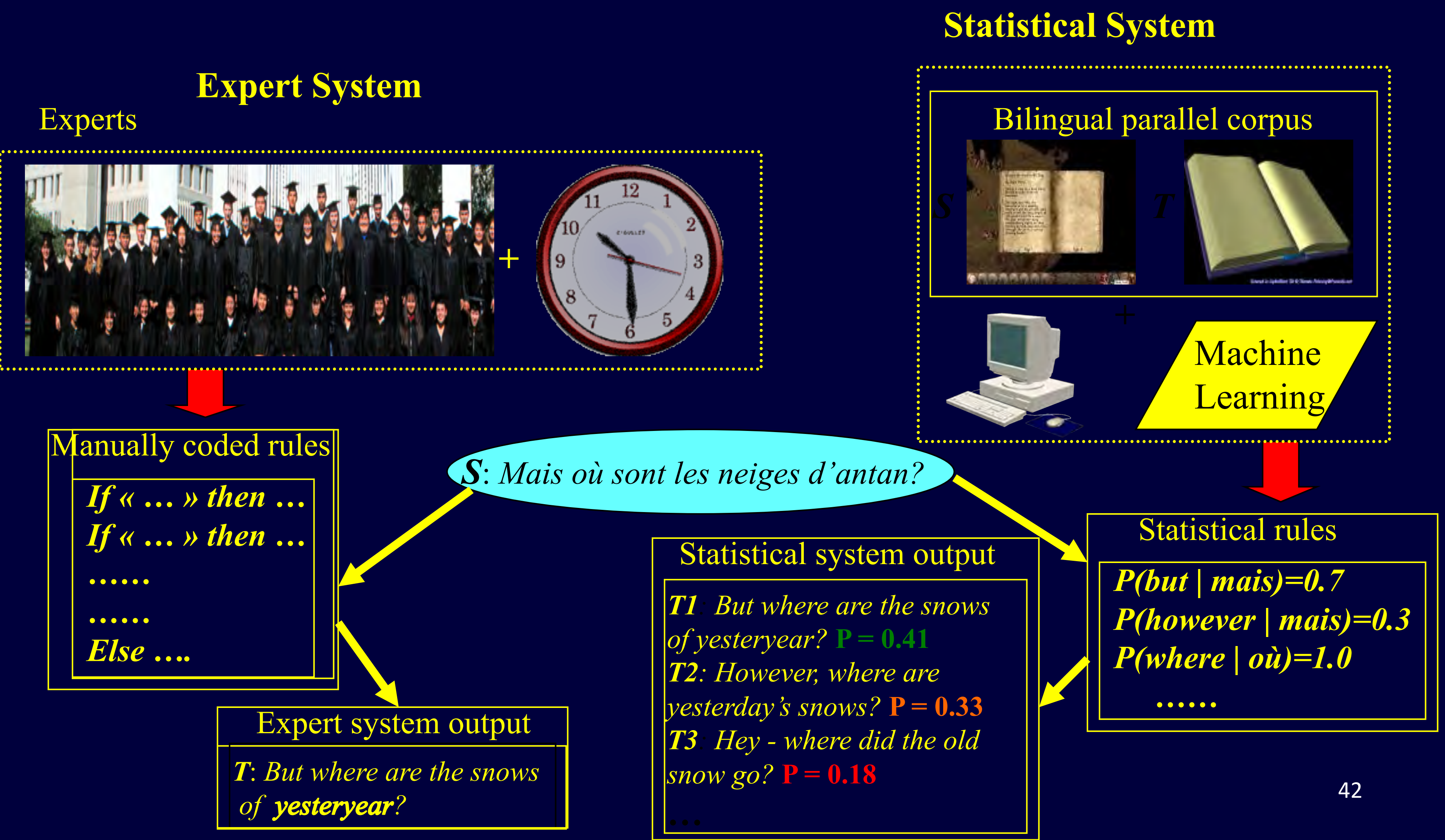
Statistical Machine Translation

- How does statistics and probabilities come into play?
  - Often statistical and rule-based MT are seen as alternatives, even opposing approaches – wrong !!!

	No Probabilities	Probabilities
Flat Structure	EBMT	SMT
Deep Structure	Transfer Interlingua	Holy Grail

  - Goal: structurally rich probabilistic models

Rule-based MT vs SMT



Statistical System

Bilingual parallel corpus

S

T

+

Machine Learning

Statistical rules

P(but | mais)=0.7
P(however | mais)=0.3
P(where | où)=1.0
……

Expert System

Experts

+

Manually coded rules

If « … » then …
If « … » then …
……
……
Else ….

S*: Mais où sont les neiges d'antan?*

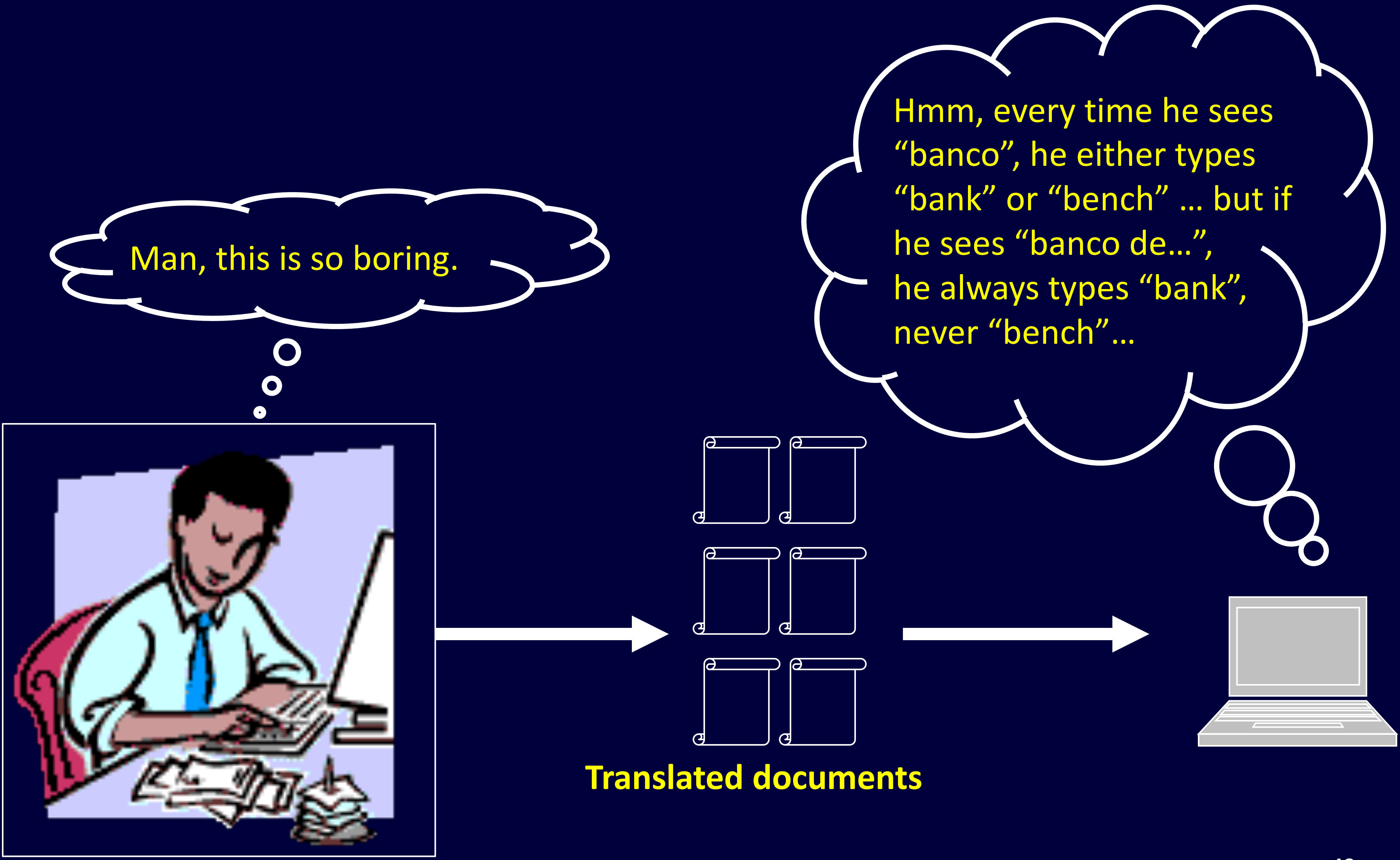
Expert system output

T*: But where are the snows of* ***yesteryear****?*

Statistical system output

T1*: But where are the snows of yesteryear?* **P = 0.41**
T2*: However, where are yesterday's snows?* **P = 0.33**
T3*: Hey - where did the old snow go?* **P = 0.18**
…

Data-Driven Machine Translation



Man, this is so boring.

Hmm, every time he sees “banco”, he either types “bank” or “bench” … but if he sees “banco de…”, he always types “bank”, never “bench”…

Translated documents

Slide by Kevin Knight

Statistical Machine Translation

- The idea is to use lots of parallel texts to model how translations are done.
  - Observe how words or groups of words are translated
  - Observe how translated words are moved around to make fluent sentences in the target sentences

Parallel Texts

1a. Garcia and associates .
1b. Garcia y asociados .

2a. Carlos Garcia has three associates .
2b. Carlos Garcia tiene tres asociados .

3a. his associates are not strong .
3b. sus asociados no son fuertes .

4a. Garcia has a company also .
4b. Garcia tambien tiene una empresa .

5a. its clients are angry .
5b. sus clientes estan enfadados .

6a. the associates are also angry .
6b. los asociados tambien estan enfadados .

7a. the clients and the associates are enemies .
7b. los clients y los asociados son enemigos .

8a. the company has three groups .
8b. la empresa tiene tres grupos .

9a. its groups are in Europe .
9b. sus grupos estan en Europa .

10a. the modern groups sell strong pharmaceuticals .
10b. los grupos modernos venden medicinas fuertes .

11a. the groups do not sell zenzanine .
11b. los grupos no venden zanzanina .

12a. the small groups are not modern .
12b. los grupos pequenos no son modernos .

Parallel Texts

Clients do not sell pharmaceuticals in Europe

→ **Clientes no venden medicinas en Europa**

1a. Garcia and associates .
1b. Garcia y asociados .

2a. Carlos Garcia has three associates .
2b. Carlos Garcia tiene tres asociados .

3a. his associates are not strong .
3b. sus asociados no son fuertes .

4a. Garcia has a company also .
4b. Garcia tambien tiene una empresa .

5a. its clients are angry .
5b. sus clientes estan enfadados .

6a. the associates are also angry .
6b. los asociados tambien estan enfadados .

7a. the clients and the associates are enemies .
7b. los clients y los asociados son enemigos .

8a. the company has three groups .
8b. la empresa tiene tres grupos .

9a. its groups are in Europe .
9b. sus grupos estan en Europa .

10a. the modern groups sell strong pharmaceuticals .
10b. los grupos modernos venden medicinas fuertes .

11a. the groups do not sell zenzanine .
11b. los grupos no venden zanzanina .

12a. the small groups are not modern .
12b. los grupos pequenos no son modernos .

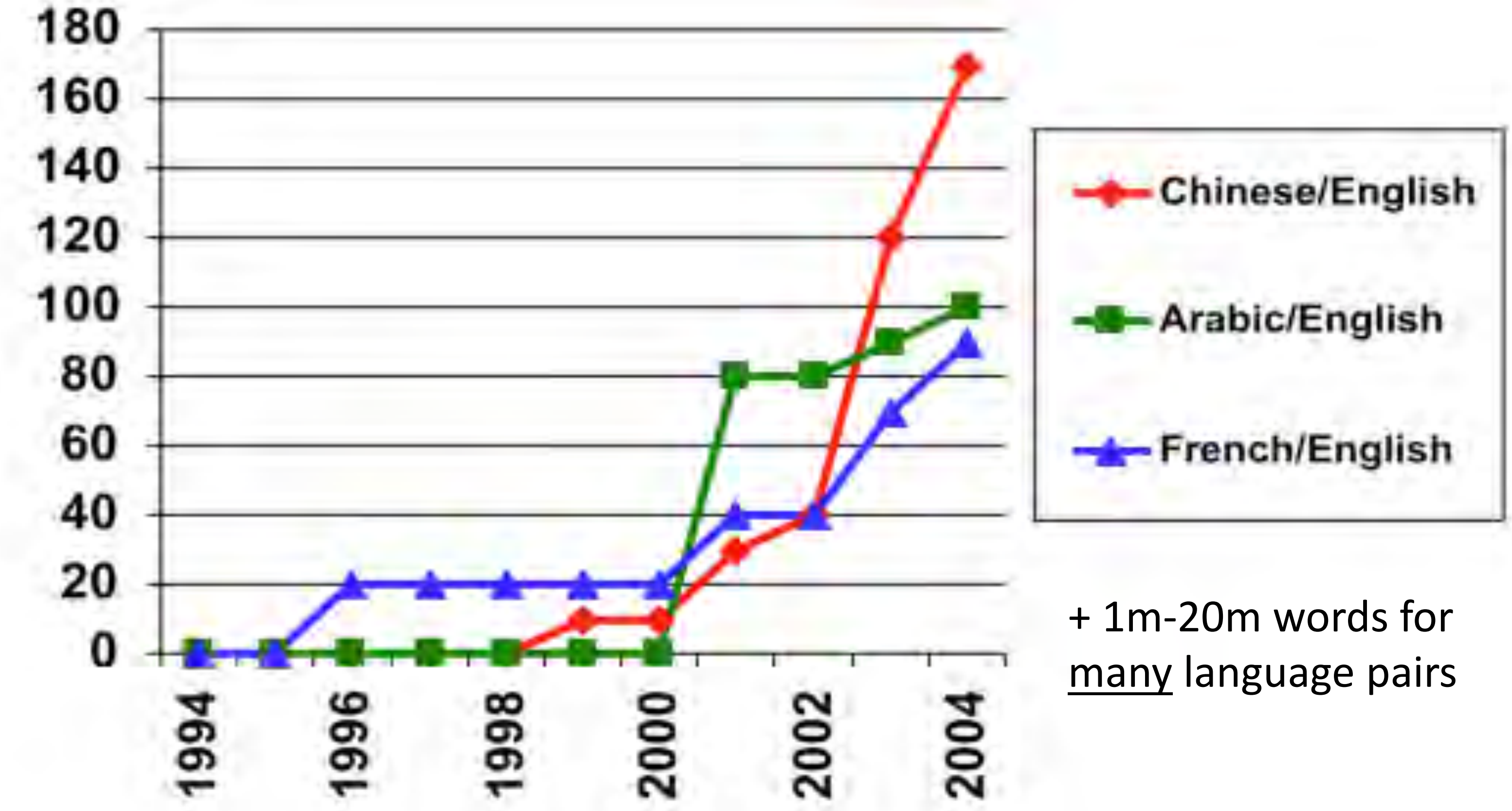
Parallel Texts

1. employment rates are very low , especially for women .
2. the overall employment rate in 2001 was 46. 8% .
3. the system covers insured employees who lose their jobs .
4. the resulting loss of income is covered in proportion to the premiums paid .
5. there has been no development in the field of disabled people .
6. overall assessment
7. no social dialogue exists in most private enterprises .
8. it should be reviewed together with all the social partners .
9. much remains to be done in the field of social protection .

1. istihdam oranları , özellikle kadınlar için çok düşüktür .
2. 2001 yılında genel istihdam oranı % 46,8' dir .
3. sistem , işini kaybeden sigortalı işsizleri kapsamaktadır .
4. ortaya çıkan gelir kaybı , ödenmiş primlerle orantılı olarak karşılanmaktadır .
5. engelli kişiler konusunda bir gelişme kaydedilmemiştir .
6. genel değerlendirme
7. özel işletmelerin çoğunda sosyal diyalog yoktur .
8. konseyin yapısı , sosyal taraflar ile birlikte yeniden gözden geçirilmelidir .
9. sosyal koruma alanında yapılması gereken çok şey vardır .

Available Parallel Data (2004)

Millions of
 words
(English side)



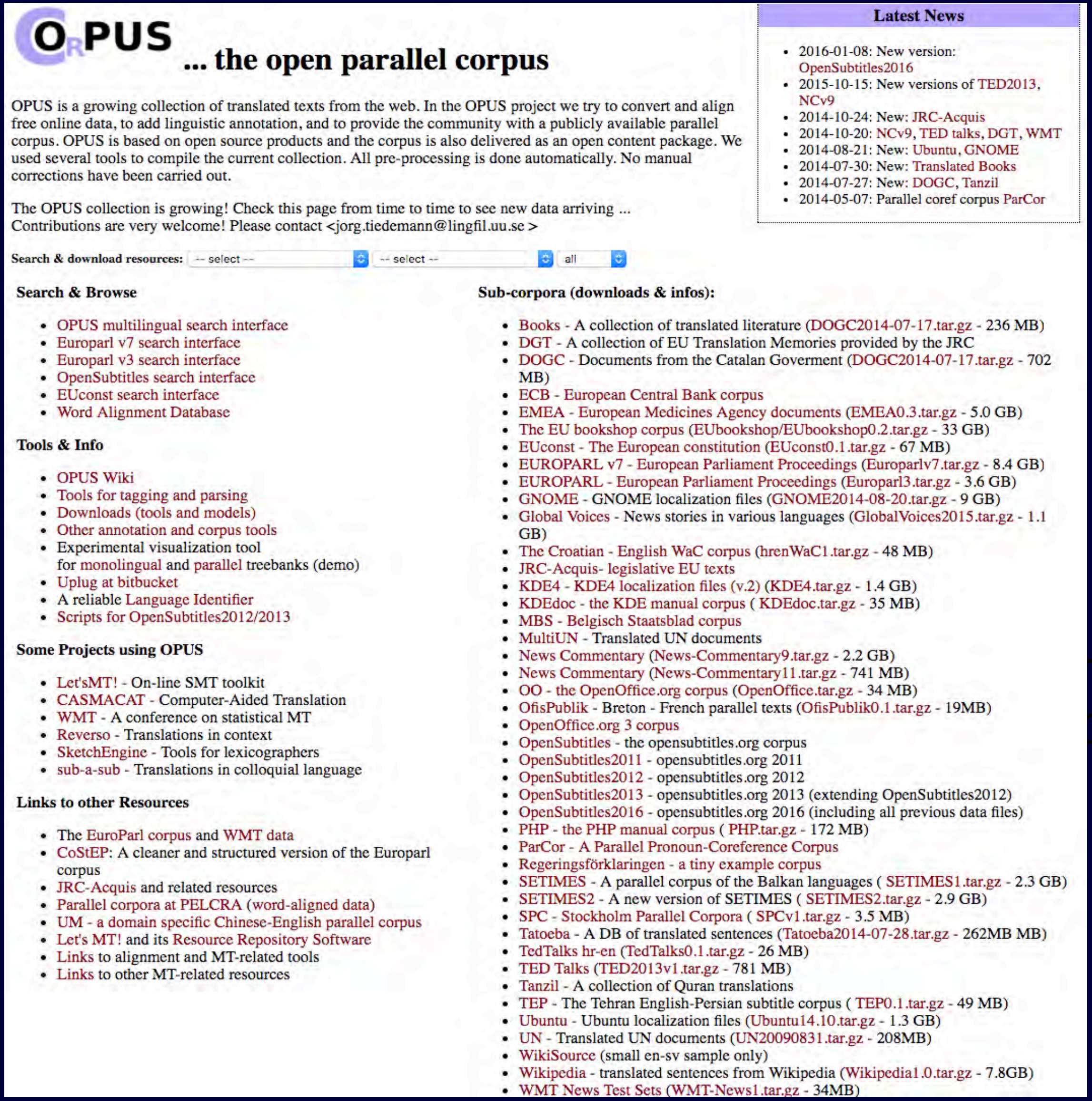
+ 1m-20m words for
many language pairs

(Data stripped of formatting, in sentence-pair format, available from the Linguistic Data Consortium at UPenn).

Available Parallel Data (2008)

- Europarl: 30 million words in 11 languages
- Acquis Communitaire: 8-50 million words in 20 EU languages
- Canadian Hansards: 20 million words from Canadian Parlimentary Proceedings
- Chinese/Arabic - English: over 100 million words from LDC
- Lots more French/English, Spanish/French/English from LDC
- Smaller corpora for many other language pairs
  - Usually English – Some other language.

Available Parallel Data (2017)



... the open parallel corpus

OPUS is a growing collection of translated texts from the web. In the OPUS project we try to convert and align free online data, to add linguistic annotation, and to provide the community with a publicly available parallel corpus. OPUS is based on open source products and the corpus is also delivered as an open content package. We used several tools to compile the current collection. All pre-processing is done automatically. No manual corrections have been carried out.

The OPUS collection is growing! Check this page from time to time to see new data arriving ...
Contributions are very welcome! Please contact <jorg.tiedemann@lingfil.uu.se >

Latest News

- 2016-01-08: New version: OpenSubtitles2016
- 2015-10-15: New versions of TED2013, NCv9
- 2014-10-24: New: JRC-Acquis
- 2014-10-20: NCv9, TED talks, DGT, WMT
- 2014-08-21: New: Ubuntu, GNOME
- 2014-07-30: New: Translated Books
- 2014-07-27: New: DOGC, Tanzil
- 2014-05-07: Parallel coref corpus ParCor

Search & download resources:

Search & Browse

- OPUS multilingual search interface
- Europarl v7 search interface
- Europarl v3 search interface
- OpenSubtitles search interface
- EUconst search interface
- Word Alignment Database

Tools & Info

- OPUS Wiki
- Tools for tagging and parsing
- Downloads (tools and models)
- Other annotation and corpus tools
- Experimental visualization tool for monolingual and parallel treebanks (demo)
- Uplug at bitbucket
- A reliable Language Identifier
- Scripts for OpenSubtitles2012/2013

Some Projects using OPUS

- Let'sMT! - On-line SMT toolkit
- CASMACAT - Computer-Aided Translation
- WMT - A conference on statistical MT
- Reverso - Translations in context
- SketchEngine - Tools for lexicographers
- sub-a-sub - Translations in colloquial language

Links to other Resources

- The EuroParl corpus and WMT data
- CoStEP: A cleaner and structured version of the Europarl corpus
- JRC-Acquis and related resources
- Parallel corpora at PELCRA (word-aligned data)
- UM - a domain specific Chinese-English parallel corpus
- Let's MT! and its Resource Repository Software
- Links to alignment and MT-related tools
- Links to other MT-related resources

Sub-corpora (downloads & infos):

- Books - A collection of translated literature (DOGC2014-07-17.tar.gz - 236 MB)
- DGT - A collection of EU Translation Memories provided by the JRC
- DOGC - Documents from the Catalan Goverment (DOGC2014-07-17.tar.gz - 702 MB)
- ECB - European Central Bank corpus
- EMEA - European Medicines Agency documents (EMEA0.3.tar.gz - 5.0 GB)
- The EU bookshop corpus (EUbookshop/EUbookshop0.2.tar.gz - 33 GB)
- EUconst - The European constitution (EUconst0.1.tar.gz - 67 MB)
- EUROPARL v7 - European Parliament Proceedings (Europarlv7.tar.gz - 8.4 GB)
- EUROPARL - European Parliament Proceedings (Europarl3.tar.gz - 3.6 GB)
- GNOME - GNOME localization files (GNOME2014-08-20.tar.gz - 9 GB)
- Global Voices - News stories in various languages (GlobalVoices2015.tar.gz - 1.1 GB)
- The Croatian - English WaC corpus (hrenWaC1.tar.gz - 48 MB)
- JRC-Acquis- legislative EU texts
- KDE4 - KDE4 localization files (v.2) (KDE4.tar.gz - 1.4 GB)
- KDEdoc - the KDE manual corpus (KDEdoc.tar.gz - 35 MB)
- MBS - Belgisch Staatsblad corpus
- MultiUN - Translated UN documents
- News Commentary (News-Commentary9.tar.gz - 2.2 GB)
- News Commentary (News-Commentary11.tar.gz - 741 MB)
- OO - the OpenOffice.org corpus (OpenOffice.tar.gz - 34 MB)
- OfisPublik - Breton - French parallel texts (OfisPublik0.1.tar.gz - 19MB)
- OpenOffice.org 3 corpus
- OpenSubtitles - the opensubtitles.org corpus
- OpenSubtitles2011 - opensubtitles.org 2011
- OpenSubtitles2012 - opensubtitles.org 2012
- OpenSubtitles2013 - opensubtitles.org 2013 (extending OpenSubtitles2012)
- OpenSubtitles2016 - opensubtitles.org 2016 (including all previous data files)
- PHP - the PHP manual corpus (PHP.tar.gz - 172 MB)
- ParCor - A Parallel Pronoun-Coreference Corpus
- Regeringsförklaringen - a tiny example corpus
- SETIMES - A parallel corpus of the Balkan languages (SETIMES1.tar.gz - 2.3 GB)
- SETIMES2 - A new version of SETIMES (SETIMES2.tar.gz - 2.9 GB)
- SPC - Stockholm Parallel Corpora (SPCv1.tar.gz - 3.5 MB)
- Tatoeba - A DB of translated sentences (Tatoeba2014-07-28.tar.gz - 262MB MB)
- TedTalks hr-en (TedTalks0.1.tar.gz - 26 MB)
- TED Talks (TED2013v1.tar.gz - 781 MB)
- Tanzil - A collection of Quran translations
- TEP - The Tehran English-Persian subtitle corpus (TEP0.1.tar.gz - 49 MB)
- Ubuntu - Ubuntu localization files (Ubuntu14.10.tar.gz - 1.3 GB)
- UN - Translated UN documents (UN20090831.tar.gz - 208MB)
- WikiSource (small en-sv sample only)
- Wikipedia - translated sentences from Wikipedia (Wikipedia1.0.tar.gz - 7.8GB)
- WMT News Test Sets (WMT-News1.tar.gz - 34MB)

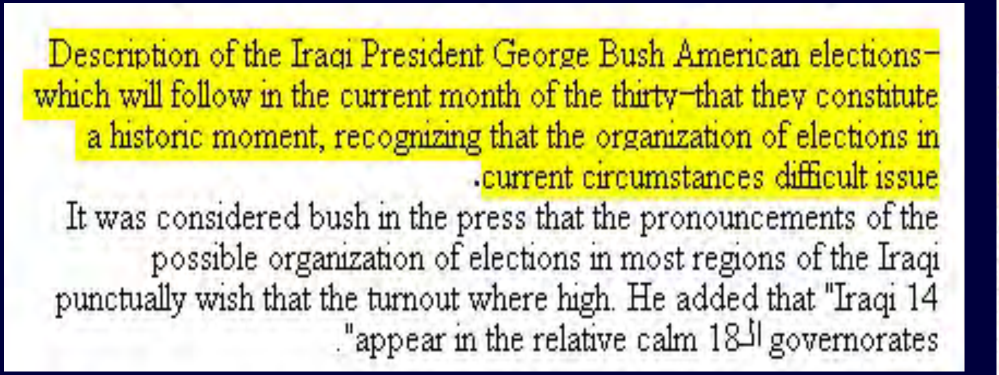
m-20m words for
ny language pairs

Available Parallel Text

- A book has a few 100,000s words
- An educated person may read 10,000 words a day
  - 3.5 million words a year
  - 300 million words a lifetime
- Soon computers will have access to more translated text than humans read in a lifetime

More data is better!

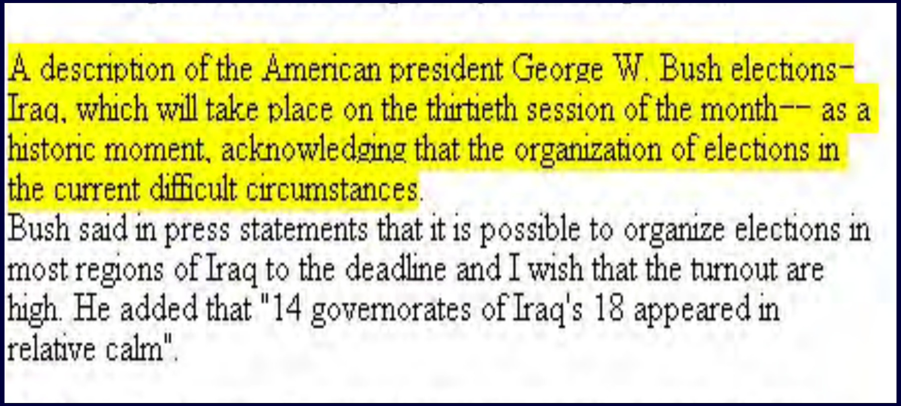
- Language Weaver Arabic to English Translation



Description of the Iraqi President George Bush American elections−which will follow in the current month of the thirty−that they constitute a historic moment, recognizing that the organization of elections in .current circumstances difficult issue

It was considered bush in the press that the pronouncements of the possible organization of elections in most regions of the Iraqi punctually wish that the turnout where high. He added that "Iraqi 14 ."appear in the relative calm 18الـ governorates

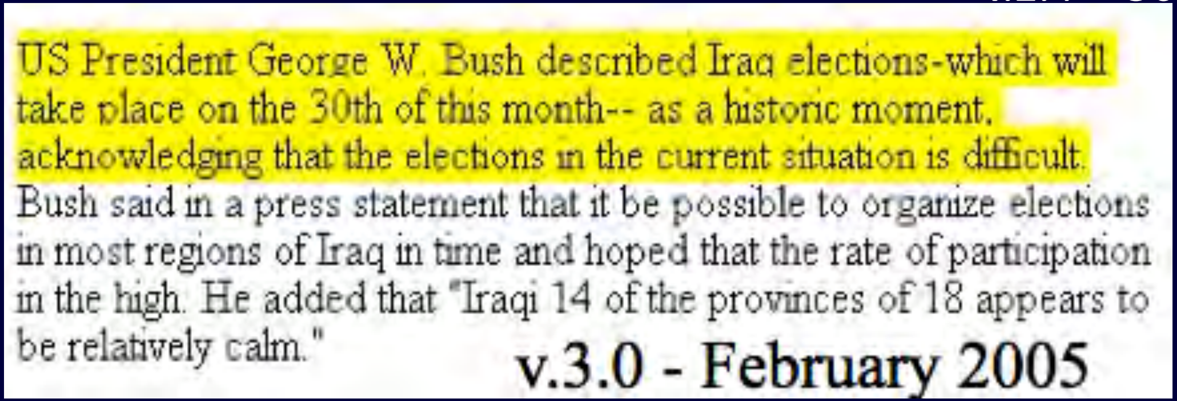
v.2.0 – October 2003



A description of the American president George W. Bush elections− Iraq, which will take place on the thirtieth session of the month-- as a historic moment, acknowledging that the organization of elections in the current difficult circumstances.

Bush said in press statements that it is possible to organize elections in most regions of Iraq to the deadline and I wish that the turnout are high. He added that "14 governorates of Iraq's 18 appeared in relative calm".

v.2.4 – October 2004



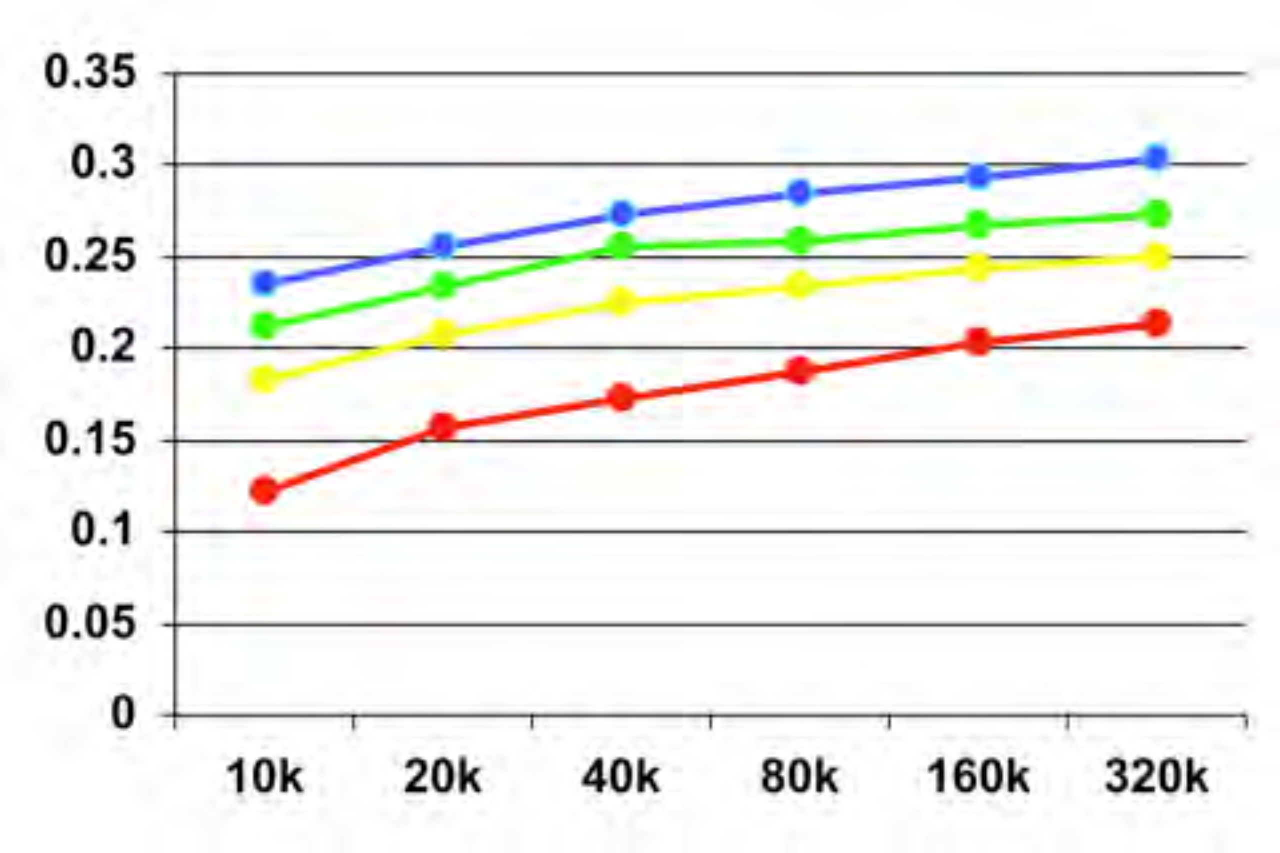
US President George W. Bush described Iraq elections-which will take place on the 30th of this month-- as a historic moment, acknowledging that the elections in the current situation is difficult.

Bush said in a press statement that it be possible to organize elections in most regions of Iraq in time and hoped that the rate of participation in the high. He added that "Iraqi 14 of the provinces of 18 appears to be relatively calm."

v.3.0 - February 2005

Sample Learning Curves

BLEU
score



Swedish/English
French/English
German/English
Finnish/English

# of sentence pairs used in training

Experiments by
Philipp Koehn

Preparing Data

- Sentence Alignment
- Tokenization/Segmentation

Sentence Alignment

The old man is happy. He has fished many times. His wife talks to him. The fish are jumping. The sharks await.

El viejo está feliz porque ha pescado muchos veces. Su mujer habla con él. Los tiburones esperan.

Sentence Alignment

1. The old man is happy.
2. He has fished many times.
3. His wife talks to him.
4. The fish are jumping.
5. The sharks await.

1. El viejo está feliz porque ha pescado muchos veces.
2. Su mujer habla con él.
3. Los tiburones esperan.

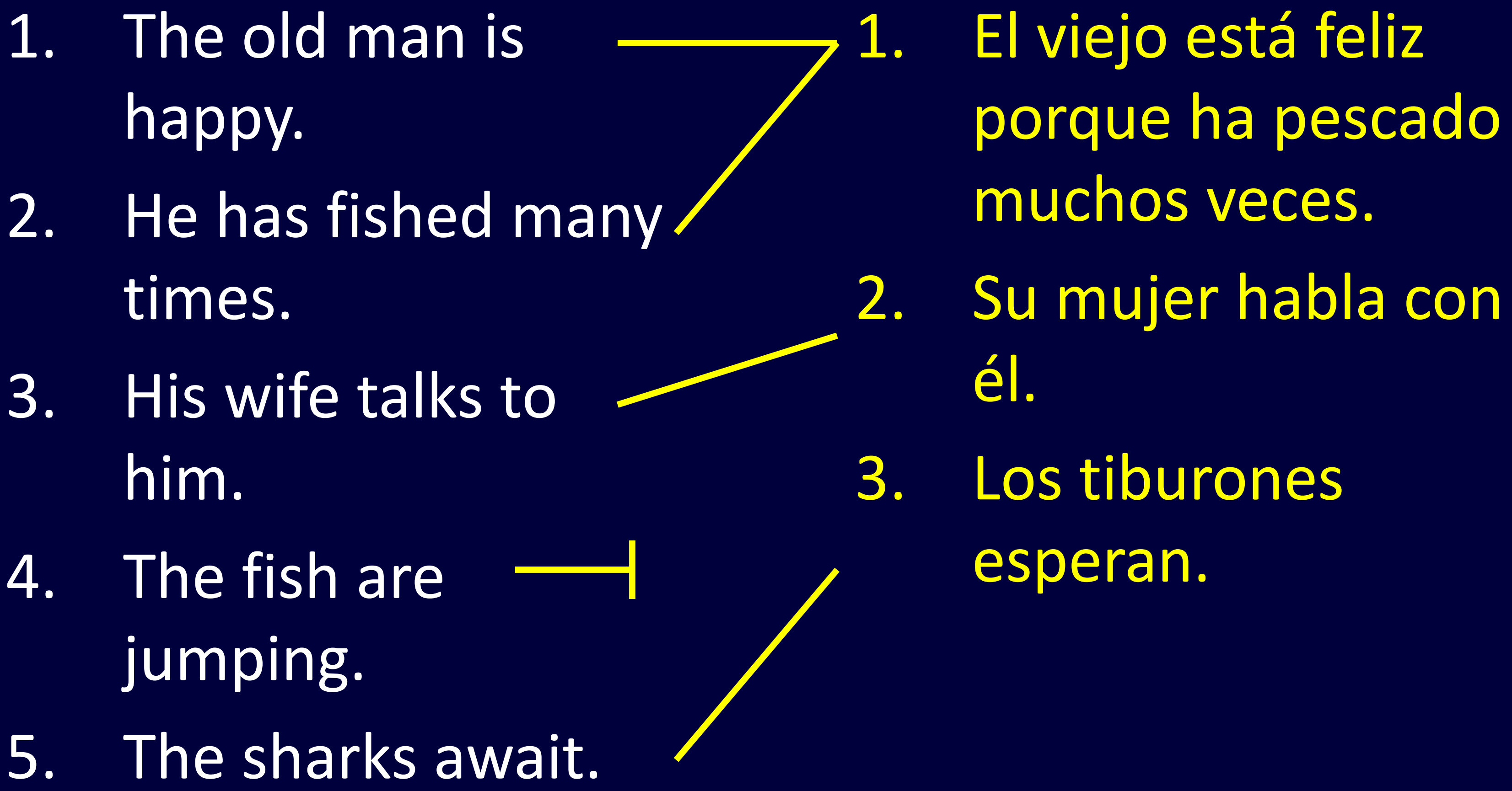
Sentence Alignment

- 1-1 Alignment
  - 1 sentence in one side aligns to 1 sentence in the other side
- 0-n, n-0 Alignment
  - A sentence in one side aligns to no sentences on the other side
- n-m Alignment (n,m>0 but typically very small)
  - n sentences on one side align to m sentences on the other side

Sentence Alignment

- Sentence alignments are typically done by dynamic programming algorithms
  - Almost always, the alignments are monotonic.
  - The lengths of sentences and their translations (mostly) correlate.
  - Tokens like numbers, dates, proper names, cognates help anchor sentences..

Sentence Alignment



1. The old man is happy.
2. He has fished many times.
3. His wife talks to him.
4. The fish are jumping.
5. The sharks await.

1. El viejo está feliz porque ha pescado muchos veces.
2. Su mujer habla con él.
3. Los tiburones esperan.

Sentence Alignment

1. The old man is happy. He has fished many times.	1. El viejo está feliz porque ha pescado muchos veces.
2. His wife talks to him.	2. Su mujer habla con él.
3. The sharks await.	3. Los tiburones esperan.

Unaligned sentences are thrown out, and sentences are merged in n-to-m alignments (n, m > 0).

Tokenization (or Segmentation)

- English
  - Input (some byte stream):

```
"There," said Bob.
```

  - Output (7 “tokens” or “words”):

```
" There , " said Bob .
```

- Chinese
  - Input (byte stream):

美国关岛国际机场及其办公室均接获
一名自称沙地阿拉伯富商拉登等发出
的电子邮件。

  - Output:

美国 关岛国 际机 场 及其 办公
室均接获 一名 自称 沙地 阿拉 伯
富 商拉登 等发 出 的 电子邮件。

The Basic Formulation of SMT

- Given a source language sentence s, what is the target language text t, that maximizes

$$p(t \mid s)$$

- So, any target language sentence t is a “potential” translation of the source sentence s
  - But probabilities differ
  - We need that t with the highest probability of being a translation.

The Basic Formulation of SMT

- Given a source language sentence s, what is the target language text t, that maximizes

$$p(t \mid s)$$

- We denote this computation as a search

$$t^* = argmax_t \, p(t \mid s)$$

The Basic Formulation of SMT

- We need to compute $t^* = argmax_t \, p(t \mid s)$

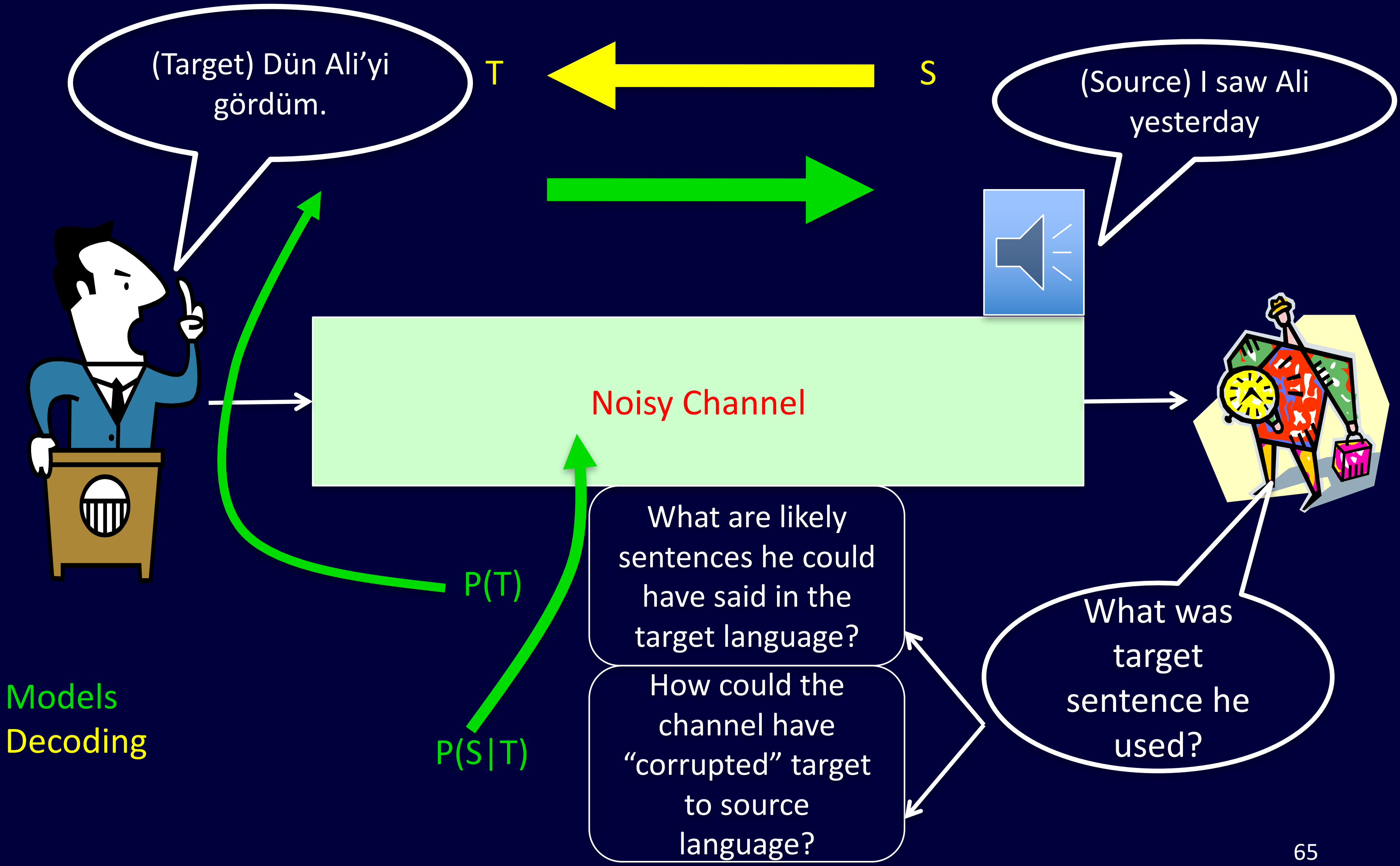
- Using Bayes' Rule we can "factorize" this into two separate problems

$$Tt^* = argmax_t \frac{p(s|t)p(t)}{p(s)}$$

$$= argmax_t \, p(s|t)p(t)$$

  – Search over all possible target sentences t
    - For a given s, *p(s)* is constant, so no need to consider it in the maximization

The Noisy Channel Model



(Target) Dün Ali'yi gördüm.

T

S

(Source) I saw Ali yesterday

Noisy Channel

P(T)

P(S|T)

What are likely sentences he could have said in the target language?

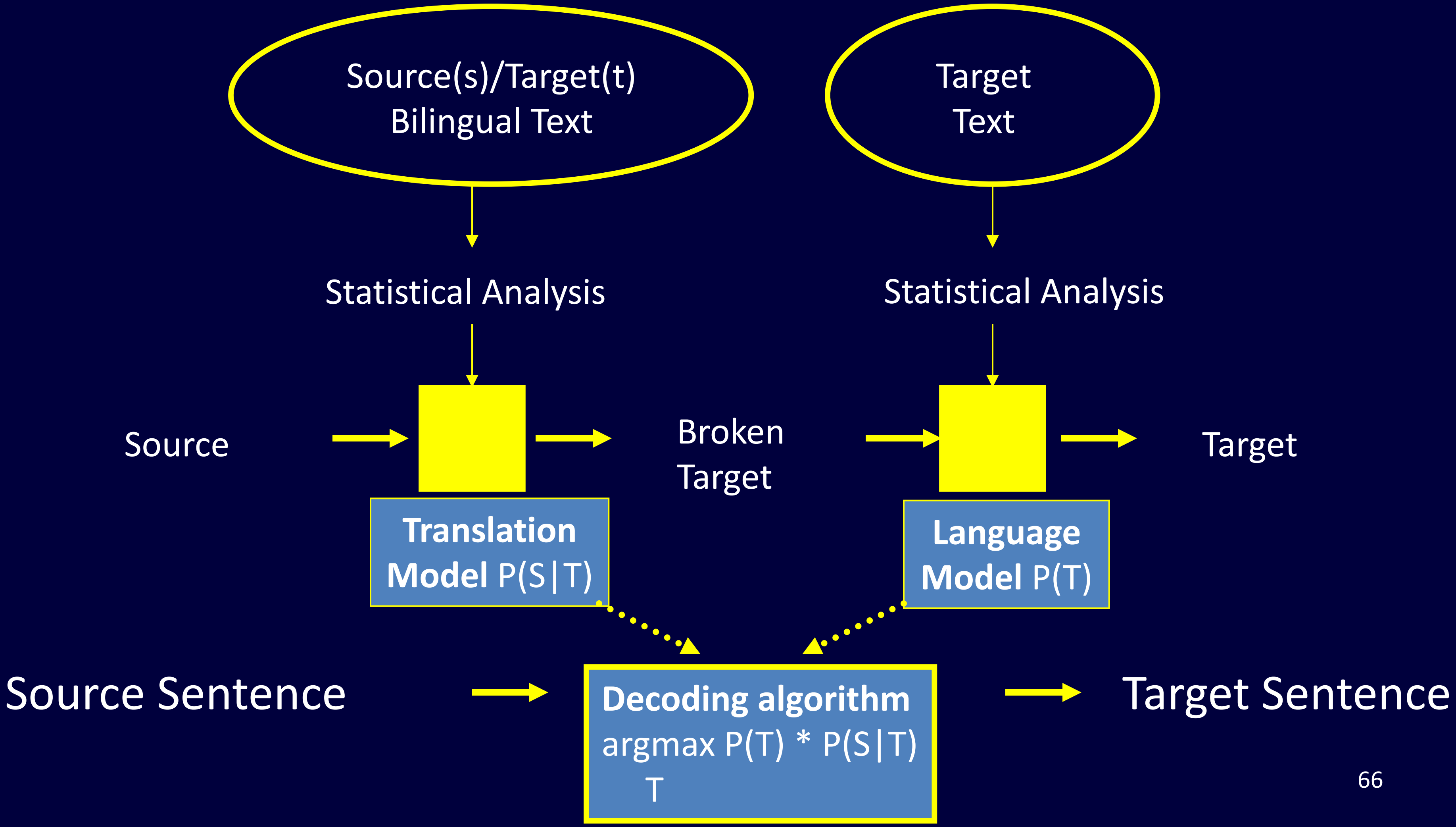
How could the channel have "corrupted" target to source language?

What was target sentence he used?

Models

Decoding

Where do the probabilities come from?



Source(s)/Target(t) Bilingual Text

Target Text

Statistical Analysis

Statistical Analysis

Source

Broken Target

Target

Translation Model P(S|T)

Language Model P(T)

Source Sentence

Decoding algorithm
argmax P(T) * P(S|T)
T

Target Sentence

The Statistical Models

- Translation model $p(S|T)$
  - Essentially models Adequacy without having to worry about Fluency.
    - $P(S|T)$ is high for sentences S, if words in S are in general translations of words in T.
- Target Language Model $p(T)$
  - Essentially models Fluency without having to worry about Adequacy
    - $P(T)$ is high if a sentence T is a fluent sentence in the target language

How do the models interact?

- Maximizing p(S | T) P(T)
  - *p(T)* models “good” target sentences (Target Language Model)
  - *p(S|T)* models whether words in source sentence are “good” translation of words in the target sentence (Translation Model)

I saw Ali yesterday	Good Target? P(T)	Good match to Source ? P(S|T)	Overall
Bugün Ali’ye gittim	🟩	🟥	🟥
Okulda kalmışlar	🟩	🟥	🟥
Var gelmek ben	🟥	🟥	🟥
Dün Ali’yi gördüm	🟩	🟩	🟩
Gördüm ben dün Ali’yi	🟩	🟩	🟩
Dün Ali’ye gördüm	🟫	🟧	🟫

Three Problems for Statistical MT

- Language model
  - Given a target sentence T, assigns p(T)
    - good target sentence -> high p(T)
    - word salad -> low p(T)

- Translation model
  - Given a pair of strings <S,T>, assigns p(S | T)
    - <S,T> look like translations -> high p(S | T)
    - <S,T> don’t look like translations -> low p(S | T)

- Decoding algorithm
  - Given a language model, a translation model, and a new sentence S … find translation T maximizing p(T) * p(S|T)

The Classic Language Model: Word n-grams

- Helps us choose among sentences
  - He is on the soccer field
  - He is in the soccer field

  - Is table the on cup the
  - The cup is on the table

  - Rice shrine
  - American shrine
  - Rice company
  - American company

The Classic Language Model

- What is a “good” target sentence? (HLT Workshop 3)
- $T = t_1\ t_2\ t_3 \ldots t_n$;
- We want P(T) to be “high”
- A good approximation is by short n-grams
  - $P(T) \cong P(t_1|START) \bullet P(t_2|START,t_1) \bullet P(t_3|t_1,t_2) \bullet \ldots \bullet P(t_i|t_{i-2},t_{i-1}) \bullet \ldots \bullet P(t_n|t_{n-2},t_{n-1})$

  - Estimate from large amounts of text
    - Maximum-likelihood estimation
    - Smoothing for unseen data
      - You can never see all of language
    - There is no data like more data (e.g., 10^9 words would be nice)

The Classic Language Model

- If the target language is English. using 2-grams

P(I saw water on the table) ≅

P(I | START) *
P(saw | I) *
P(water | saw) *
P(on | water) *
P(the | on) *
P(table | the) *
P(END | table)

The Classic Language Model

- If the target language is English, using 3-grams

P(I saw water on the table) ≅

P(I | START, START) *

P(saw | START, I) *

P(water | I, saw) *

P(on | saw, water) *

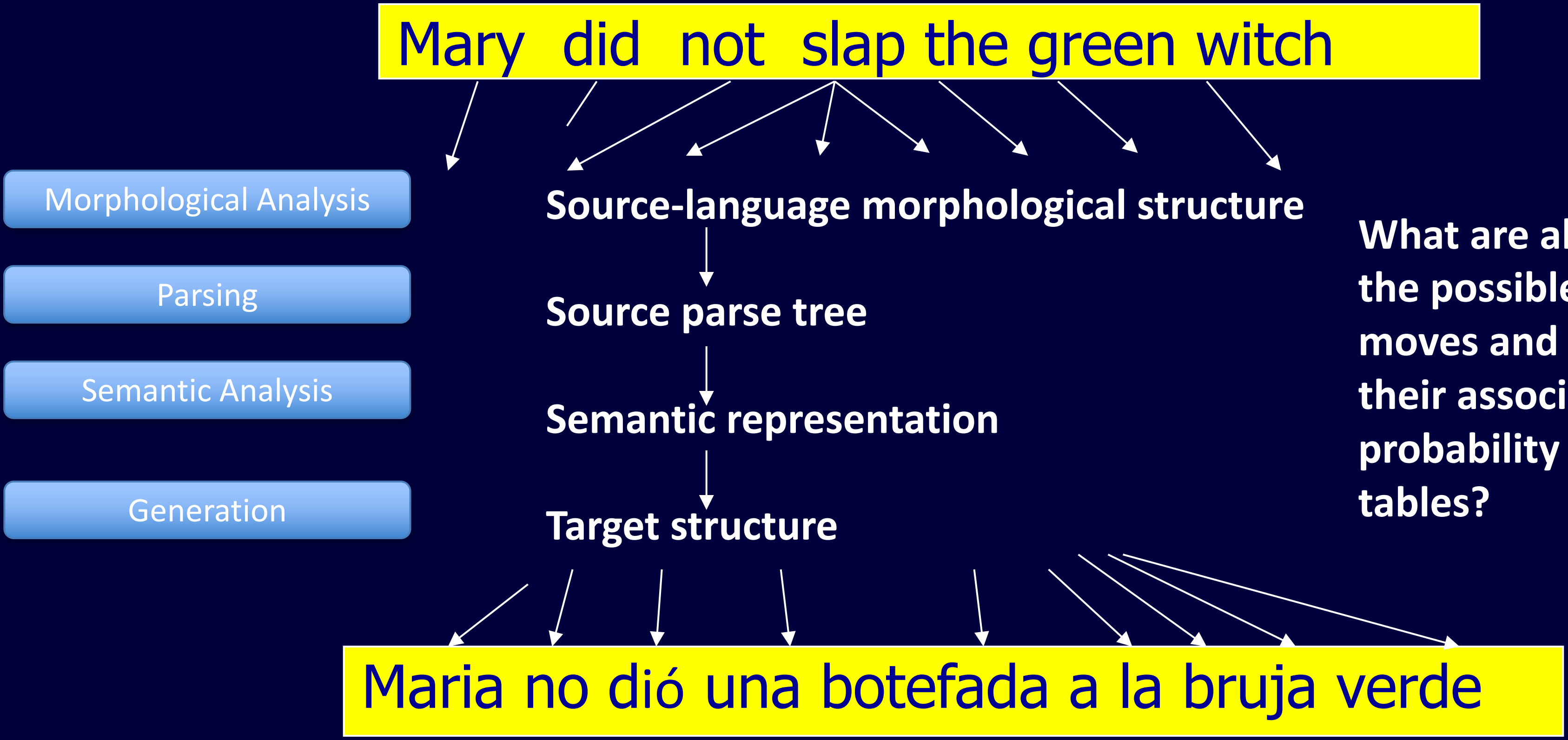
P(the | water, on) *

P(table | on, the) *

P(END | the, table)

Translation Model?

Generative approach:



Mary did not slap the green witch

Morphological Analysis

Source-language morphological structure

Parsing

Source parse tree

Semantic Analysis

Semantic representation

Generation

Target structure

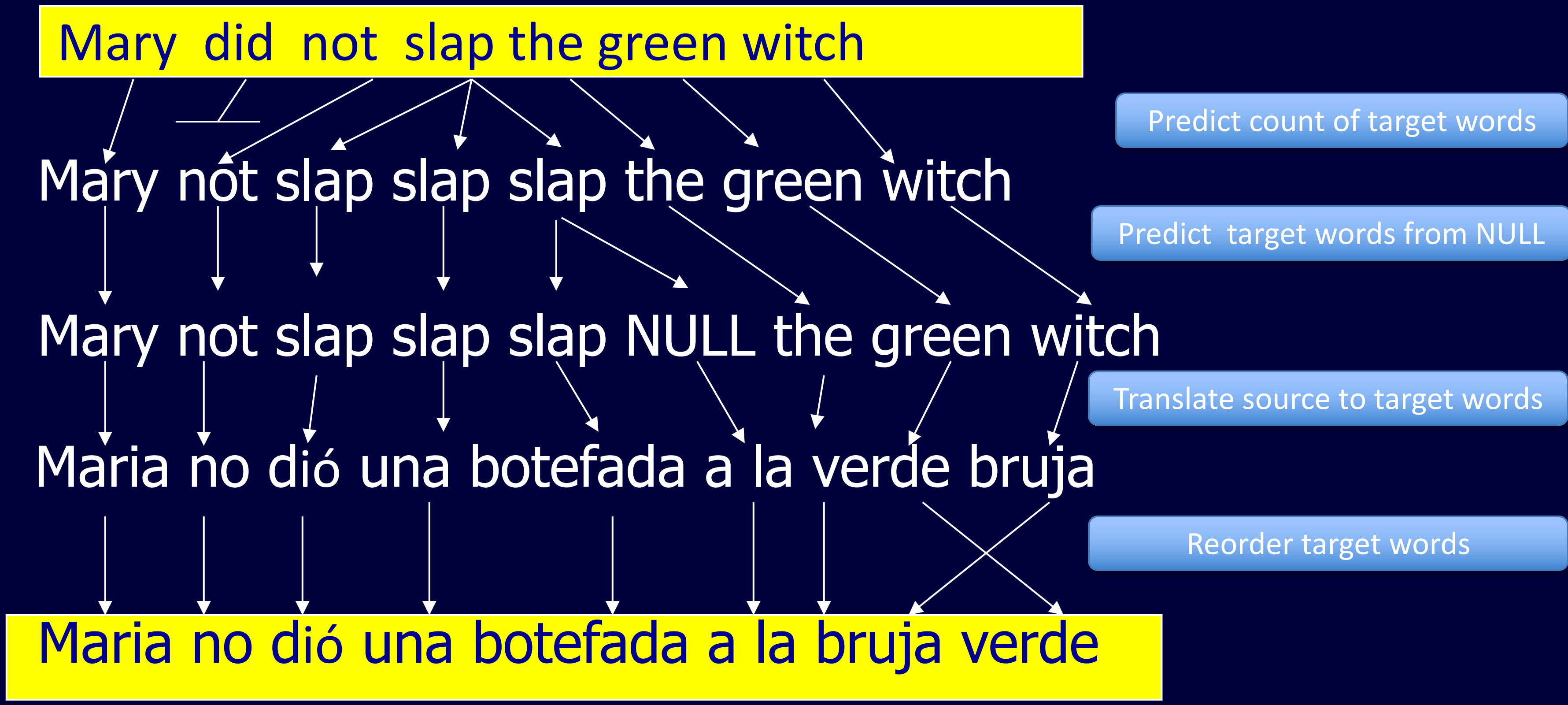
Maria no dió una botefada a la bruja verde

What are all the possible moves and their associated probability tables?

The Classic Translation Model

Word Substitution/Permutation [IBM Model 3, Brown et al., 1993]

Generative approach:



Mary did not slap the green witch

Predict count of target words

Mary not slap slap slap the green witch

Predict target words from NULL

Mary not slap slap slap NULL the green witch

Translate source to target words

Maria no dió una botefada a la verde bruja

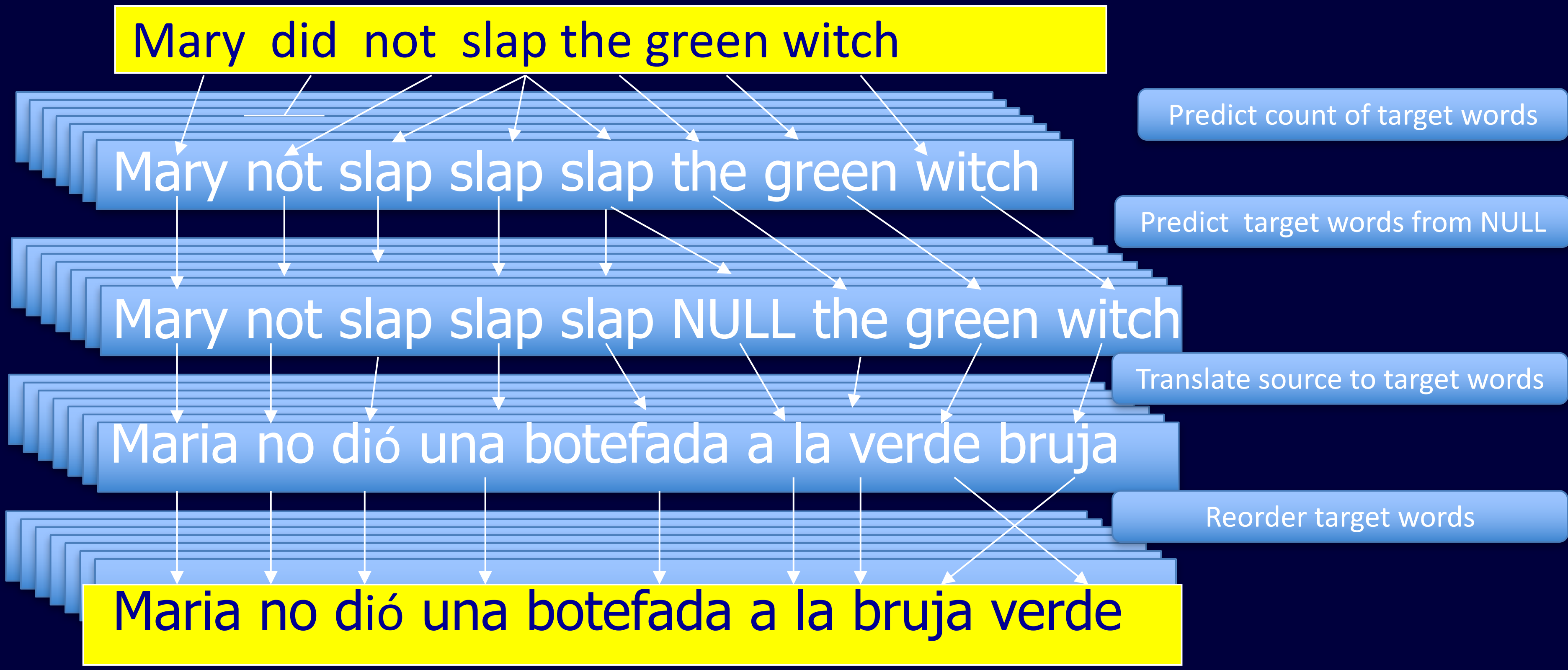
Reorder target words

Maria no dió una botefada a la bruja verde

The Classic Translation Model

Word Substitution/Permutation [IBM Model 3, Brown et al., 1993]

Generative approach:



Mary did not slap the green witch

Mary not slap slap slap the green witch

Mary not slap slap slap NULL the green witch

Maria no dió una botefada a la verde bruja

Maria no dió una botefada a la bruja verde

Predict count of target words

Predict target words from NULL

Translate source to target words

Reorder target words

Selected as the most likely by P(T)

Basic Translation Model (IBM M-1)

- Model $p(t \mid s, m)$
  - $t = <t_1, t_2, \ldots, t_m>$, $s = <s_1, s_2, \ldots, s_n>$
- Lexical translation makes the following assumptions
  - Each word t_i in t is generated from exactly one word in s.
  - Thus, we have a latent alignment a_i that indicates which word t_i “came from.” Specifically it came from t_{a_i}.
  - Given the alignments a, translation decisions are conditionally independent of each other and depend only on the aligned source word t

Basic Translation Model (IBM M-1)

$$p(t|s,m) = \underbrace{\sum_{a\in[0,n]^m} p(a\,|s,m)}_{\text{p(alignment)}} \times \underbrace{\prod_{i=1}^{m} p(t_i\,|s_{a_i})}_{\text{p(translation | alignment)}}$$

Parameters of the IBM 3 Model

- Fertility: How many words does a source word get translated to?
  - n(k | s): the probability that the source word s gets translated as k target words
  - Fertility depends solely on the source words in question and not other source words in the sentence, or their fertilities.

- Null Probability: What is the probability of a word magically appearing in the target at some position, without being the translation of any source word?
  - P-null

Parameters of the IBM 3 Model

- Translation: How do source words translate?
  - tr(t|s): the probability that the source word s gets translated as the target word t
  - Once we fix n(k | s) we generate k target words
- Reordering: How do words move around in the target sentence?
  - d(j | i): distortion probability – the probability of word at position i in a source sentence being translated as the word at position j in target sentence.
    - Very dubious!!

How IBM Model 3 works

1. For each source word s_i indexed by i = 1, 2, ..., m, choose fertility phi_i with probability $n(phi_i \mid s_i)$.
2. Choose the number phi_0 of “spurious” target words to be generated from s_0 = NULL

How IBM Model 3 works

3. Let q be the sum of fertilities for all words, including NULL.
4. For each i = 0, 1, 2, ..., m, and each k = 1, 2, ..., phi_i, choose a target word t_{ik} with probability $tr(t_{ik} \mid s_i)$.
5. For each i = 1, 2, ..., l, and each k = 1, 2, ..., phi_i, choose target position pi_{ik} with probability $d(pi_{ik} \mid i,l,m)$.

How IBM Model 3 works

6. For each k = 1, 2, ..., phi_0, choose a position pi_{0k} from the remaining vacant positions in 1, 2, ... q, for a total probability of $1/phi_0$.
7. Output the target sentence with words t_{ik} in positions pi_{ik} $(0 <= i <= m, 1 <= k <= phi_i)$.

Example

```
b c d
|   |
| +-+
| | |
x y z
```

```
b d
| |
| |
| |
x y
```

- n-parameters
- n(0,b)=0, n(1,b)=2/2=1
- n(0,c)=1/1=1, n(1,c)=0
- n(0,d)=0,n(1,d)=1/2= 0.5, n(2,d)=1/2=0.5

Example

```
b c d
|   |
| +-+
| | |
x y z
```

```
b d
| |
| |
| |
x y
```

- t-parameters
- t(x|b)=1.0
- t(y|d)=2/3
- t(z|d)=1/3

Example

```
b c d      b d
|   |      | |
| +-+      | |
| | |      | |
x y z      x y
```

- d-parameters
- d(1|1,3,3)=1.0
- d(1|1,2,2)=1.0
- d(2|2,3,3)=0.0
- d(3|3,3,3)=1.0
- d(2|2,2,2)=1.0

Example

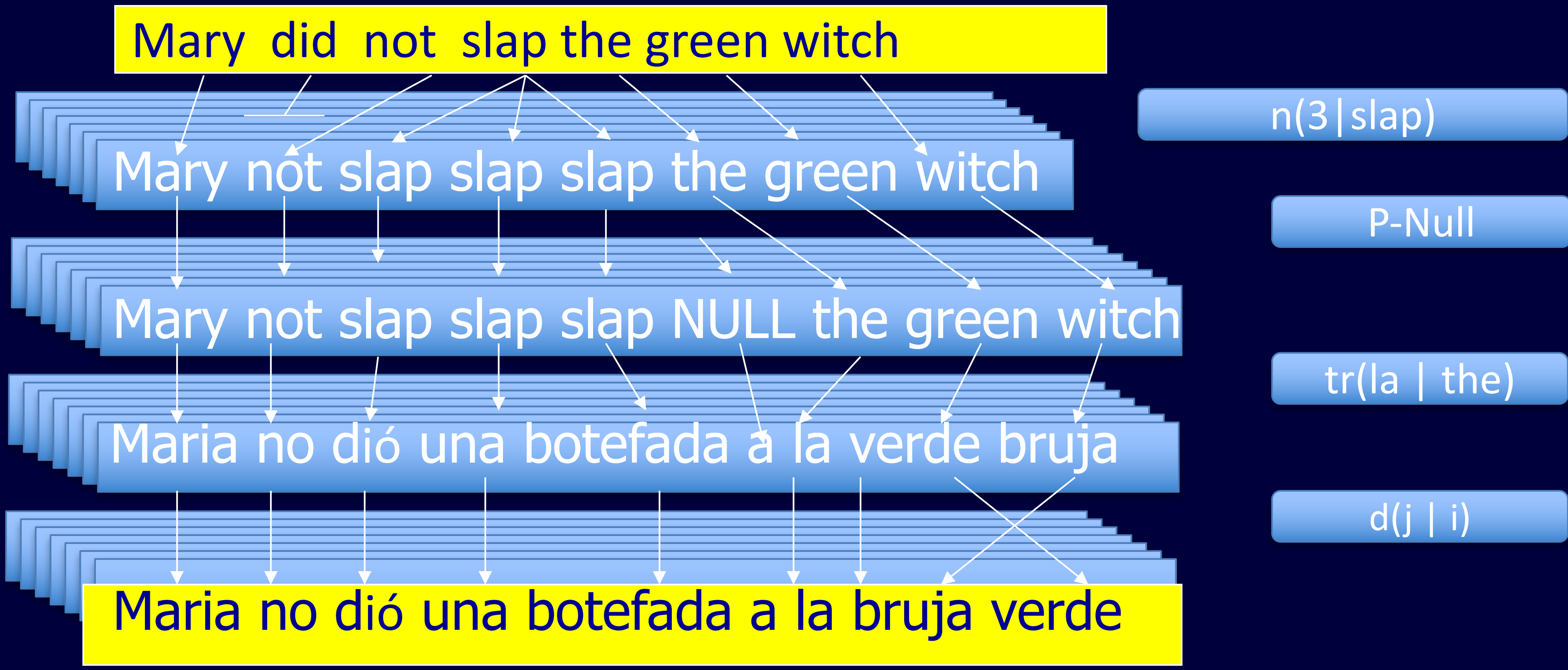
```
b c d      b d
|   |      | |
| +-+      | |
| | |      | |
x y z      x y
```

- p1
- No target words are generated by NULL so p1 = 0.0

The Classic Translation Model

Word Substitution/Permutation [IBM Model 3, Brown et al., 1993]

Generative approach:



Mary did not slap the green witch

Mary not slap slap slap the green witch

Mary not slap slap slap NULL the green witch

Maria no dió una botefada a la verde bruja

Maria no dió una botefada a la bruja verde

n(3|slap)

P-Null

tr(la | the)

d(j | i)

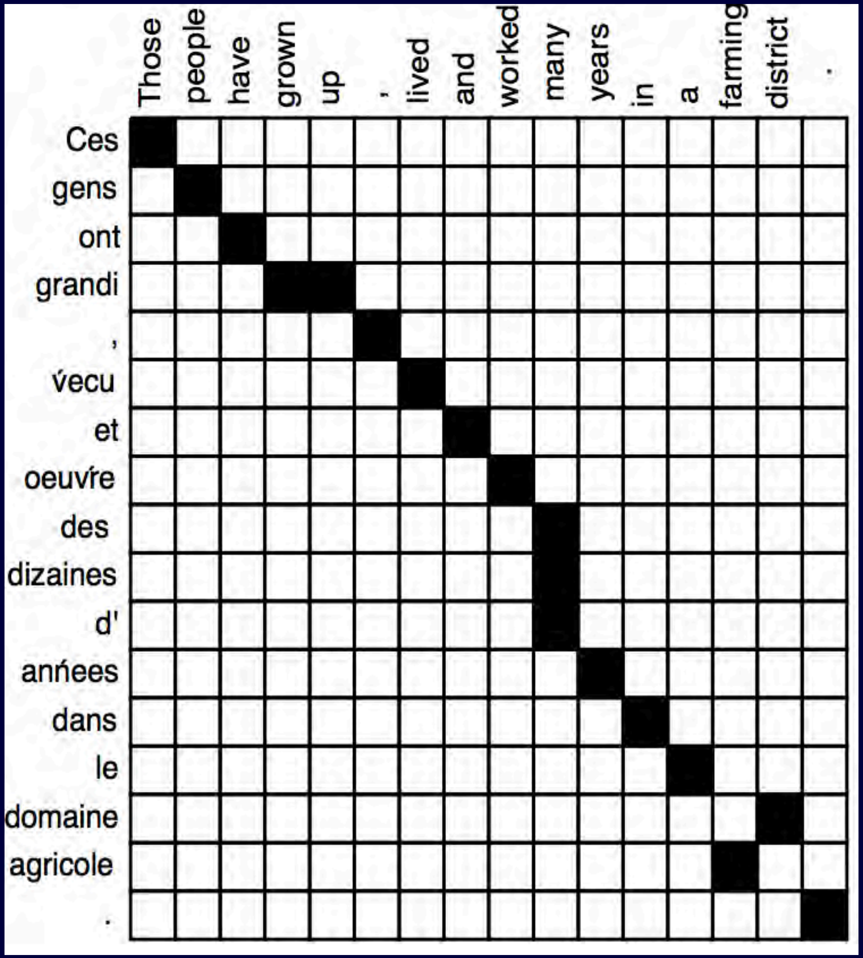
Selected as the most likely by P(T)

How do we get these parameters?

- Remember we had aligned parallel sentences

- Now we need to figure out how words align with other words.
  - Word alignment

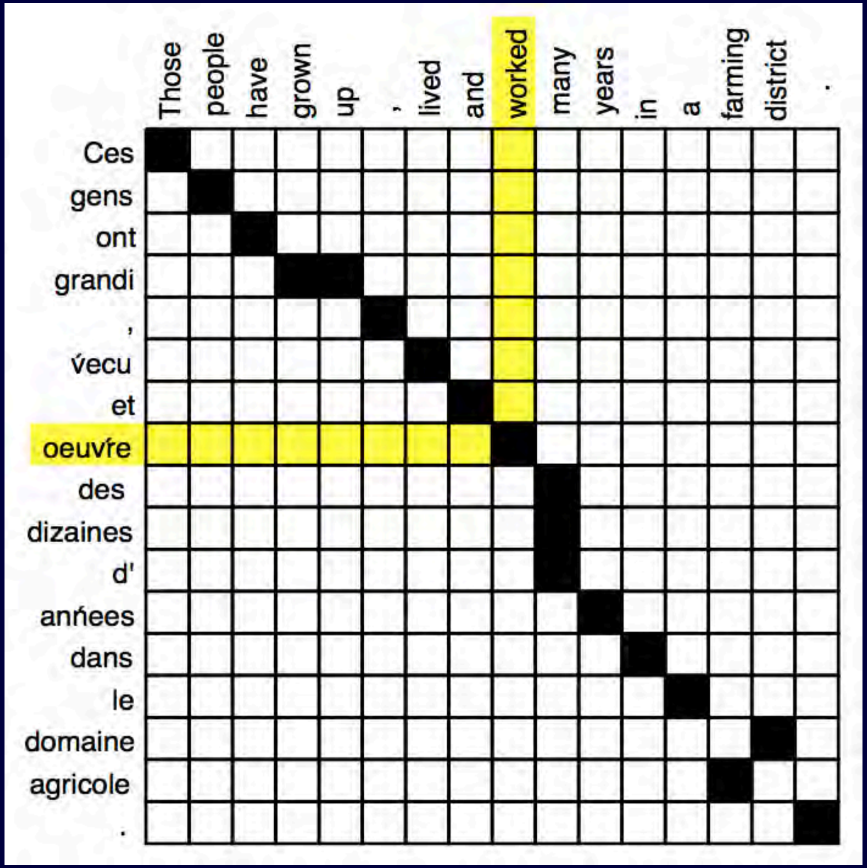
Word Alignments



	Those	people	have	grown	up	,	lived	and	worked	many	years	in	a	farming	district	.
Ces	■															
gens		■														
ont			■													
grandi				■	■											
,						■										
v́ecu							■									
et								■								
oeuvŕe									■							
des										■						
dizaines										■						
d'										■						
anńees											■					
dans												■				
le													■			
domaine															■	
agricole														■		
.																■

- One source word can map to 0 or more target words
  - – But not vice versa
    - technical reasons
- Some words in the target can magically be generated from an invisible NULL word
- A target word can only be generated from one source word
  - – technical reasons

Word Alignments



	Those	people	have	grown	up	,	lived	and	worked	many	years	in	a	farming	district	.
Ces	■															
gens		■														
ont			■													
grandi				■	■											
,						■										
v́ecu							■									
et								■								
oeuv́re									■							
des										■						
dizaines										■						
d'										■						
anńees											■					
dans												■				
le													■			
domaine															■	
agricole														■		
.																■

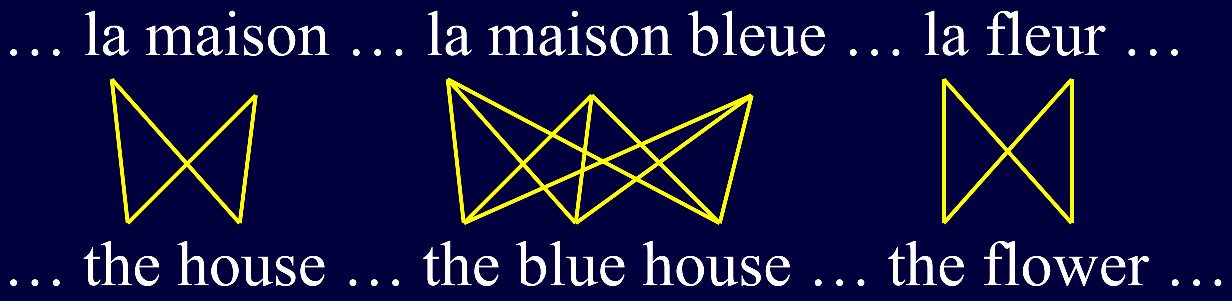
$$tr(oeuvre\ |worked) = \frac{c(oeuvre\ |worked)}{c(worked)}$$

- Count over all aligned sentences
- worked
  - **fonctionné(30), travaillé(20), marché(27), oeuvré (13)**
  - **tr(oeuvre|worked)=0.13**
- Similarly, n(3, many) can be computed.

How do we get these alignments?

- We only have aligned sentences and the constraints:
  - One source word can map to 0 or more target words
    - But not vice versa
  - Some words in the target can magically be generated from an invisible NULL word
  - A target word can only be generated from one source word
- Estimation – Maximization Algorithm
  - Mathematics is rather complicated

How do we get these alignments?



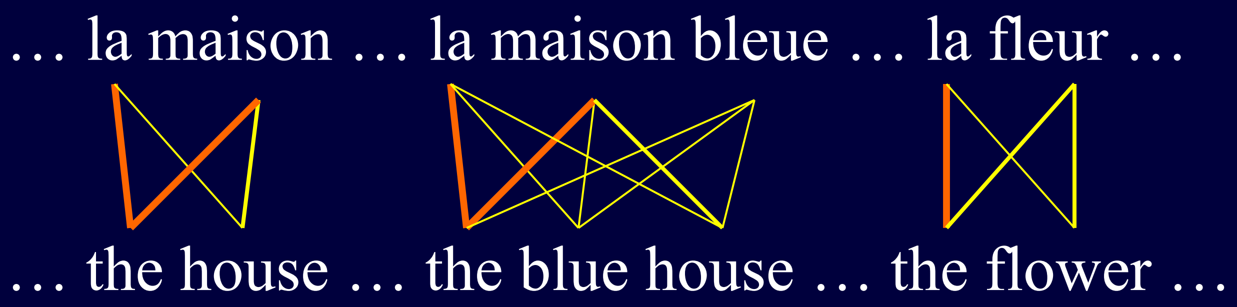
… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

All word alignments equally likely

All p(french-word | english-word) equally likely

How do we get these alignments?

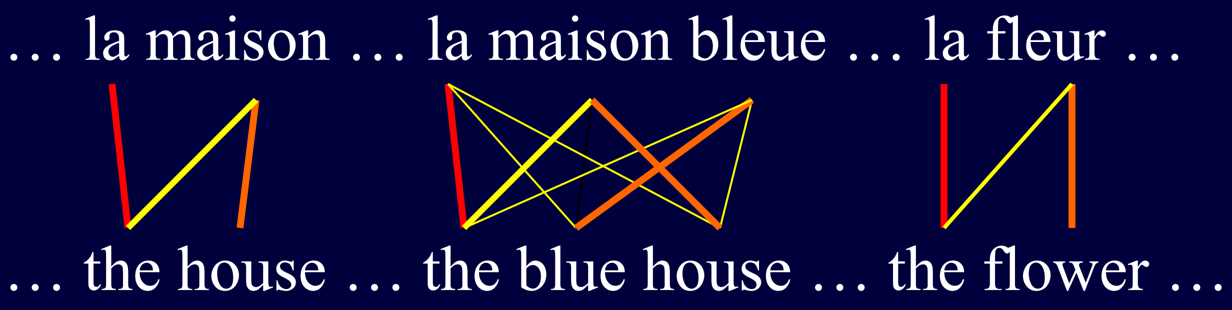


… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

“la” and “the” observed to co-occur frequently,
so p(la | the) is increased.

How do we get these alignments?



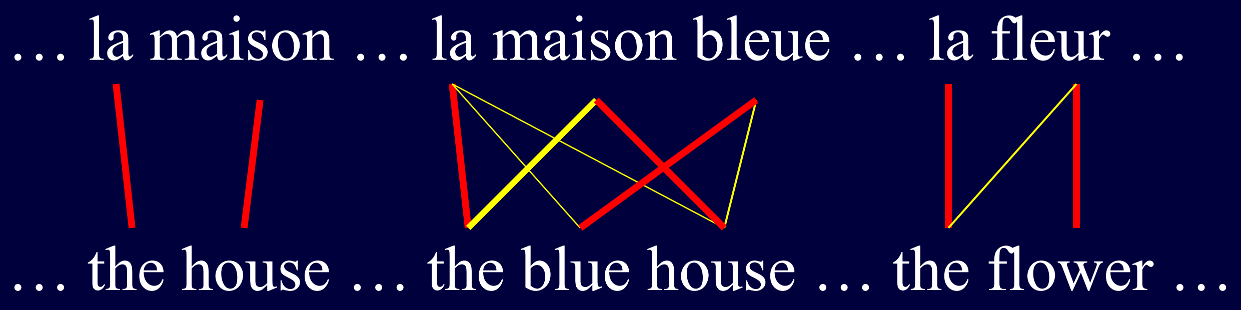
… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

“house” co-occurs with both “la” and “maison”, but p(maison | house) can be raised without limit, to 1.0, while p(la | house) is limited because of “the”

(pigeonhole principle)

How do we get these alignments?

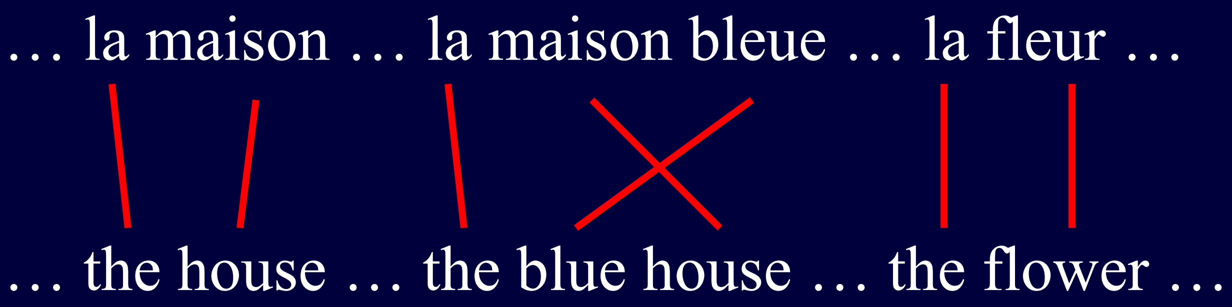


… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

settling down after another iteration

How do we get these alignments?



… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

Inherent hidden structure revealed by EM training!

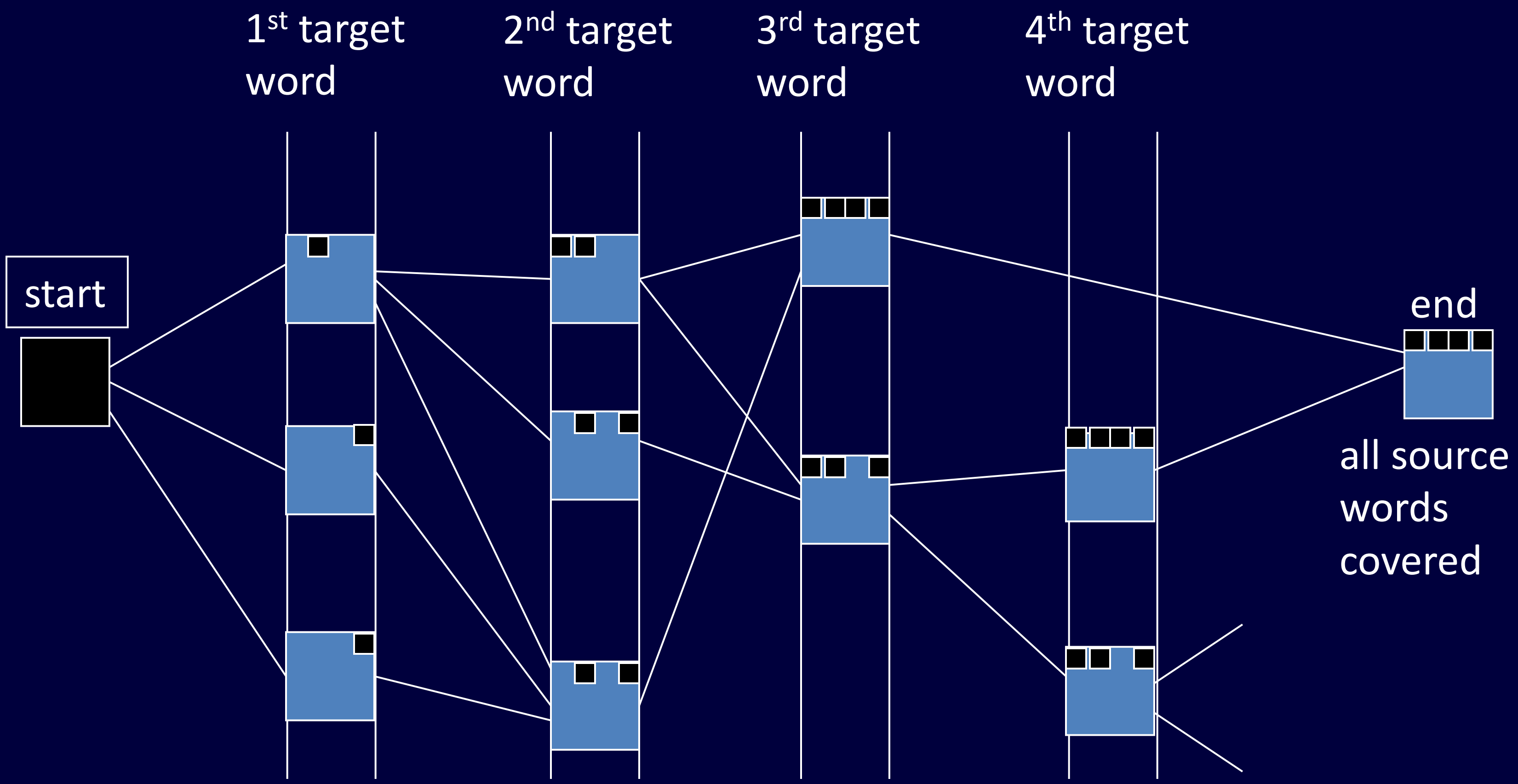
For further details, see:

- “A Statistical MT Tutorial Workbook” (Knight, 1999).
- “The Mathematics of Statistical Machine Translation” (Brown et al, 1993)
- Software: GIZA++

Decoding for "Classic" Models

- Of all conceivable English word strings, find the one maximizing $p(t) * p(t \mid s)$
- Decoding is an NP-complete challenge
  - Reduction to Traveling Salesman problem (Knight, 1999)
- Several search strategies are available
- Each potential target output is called a *hypothesis*.

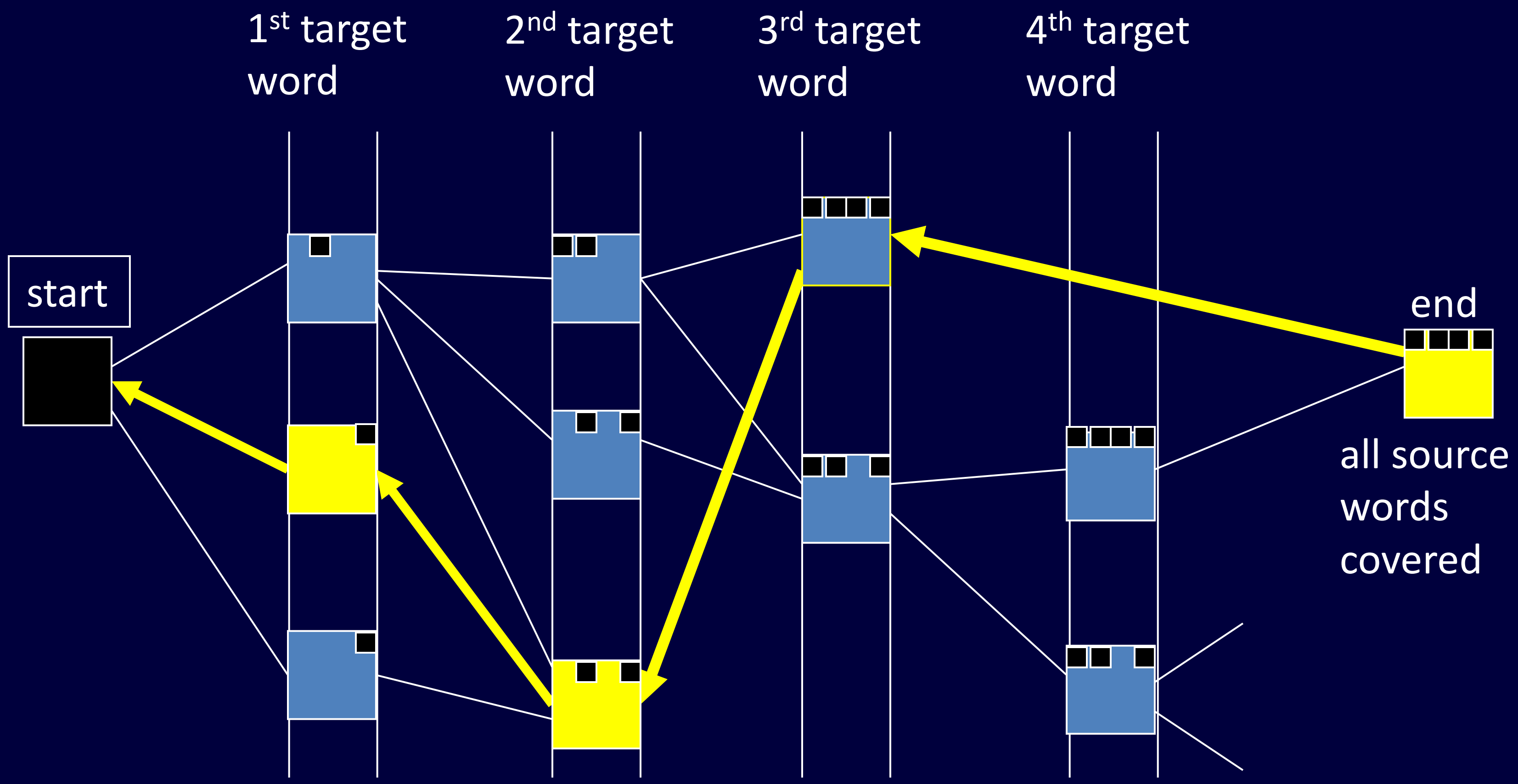
Dynamic Programming Beam Search



Each partial translation hypothesis contains:
- Last English word chosen + source words covered by it
- Next-to-last English word chosen
- Entire coverage vector (so far) of source sentence ■■ ■
- Language model and translation model scores (so far)

[Jelinek, 1969;
Brown et al, 1996 US Patent;
(Och, Ueffing, and Ney, 2001]

Dynamic Programming Beam Search



Each partial translation hypothesis contains:
- Last English word chosen + source words covered by it
- Next-to-last English word chosen
- Entire coverage vector (so far) of source sentence ■■ ■
- Language model and translation model scores (so far)

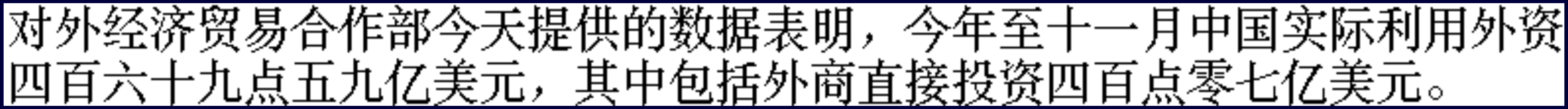
[Jelinek, 1969;
Brown et al, 1996 US Patent;
(Och, Ueffing, and Ney, 2001]

The Classic Results

- *la politique de la haine .* *(Original Source)*
- politics of hate . (Reference Translation)
- the policy of the hatred . (IBM4+N-grams+Stack)

- *nous avons signé le protocole .* *(Original Source)*
- we did sign the memorandum of agreement . (Reference Translation)
- we have signed the protocol . (IBM4+N-grams+Stack)

- *où était le plan solide ?* *(Original Source)*
- but where was the solid plan ? (Reference Translation)
- where was the economic base ? (IBM4+N-grams+Stack)

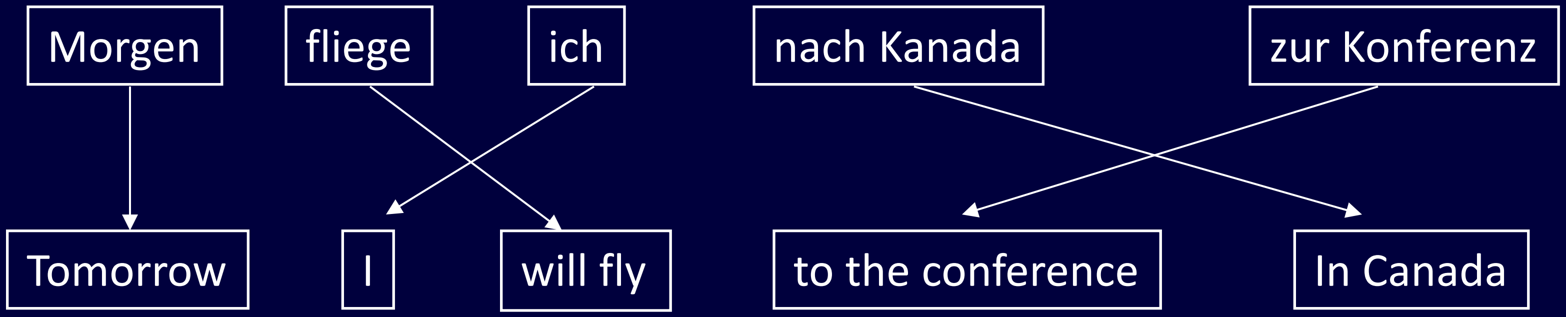

对外经济贸易合作部今天提供的数据表明，今年至十一月中国实际利用外资四百六十九点五九亿美元，其中包括外商直接投资四百点零七亿美元。

the Ministry of Foreign Trade and Economic Cooperation, including foreign
direct investment 40.007 billion US dollars today provide data include
that year to November china actually using foreign 46.959 billion US dollars and

Flaws of Word-Based MT

- Multiple source words for one target word
  - IBM models can do one-to-many (fertility) but not many-to-one
- Phrasal Translation
  - "real estate", "note that", "interest in"
- Syntactic Transformations
  - Verb at the beginning in Arabic
  - Translation model penalizes any proposed re-ordering
  - Language model not strong enough to force the verb to move to the right place

Phrase-Based Statistical MT



- Source input segmented in to phrases
  - “phrase” is any sequence of words
- Each phrase is probabilistically translated into target
  - P(to the conference | zur Konferenz)
  - P(into the meeting | zur Konferenz)
- Phrases are probabilistically re-ordered

Advantages of Phrase-Based SMT

- Many-to-many mappings can handle non-compositional phrases
- Local context is very useful for disambiguating
  - "Interest rate" → …
  - "Interest in" → …
- The more data, the longer the learned phrases
  - Sometimes whole sentences

How to Learn the Phrase Translation Table?

- One method: “alignment templates”
- Start with word alignment, build phrases from that.

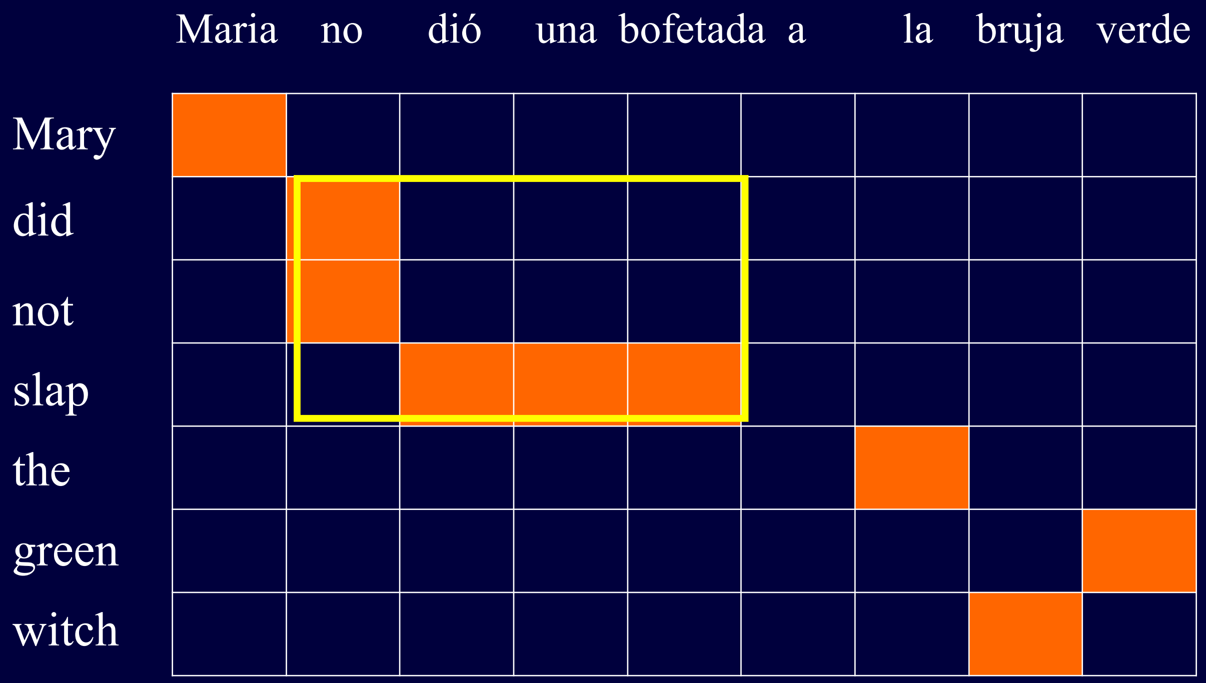
	Maria	no	dió	una	bofetada	a	la	bruja	verde
Mary	■								
did		■							
not		■							
slap			■	■	■				
the							■		
green									■
witch								■	

This word-to-word alignment is a by-product of training a translation model like IBM-Model-3.

This is the best (or “Viterbi”) alignment.

How to Learn the Phrase Translation Table?

- One method: “alignment templates” (Och et al, 1999)
- Start with word alignment, build phrases from that.



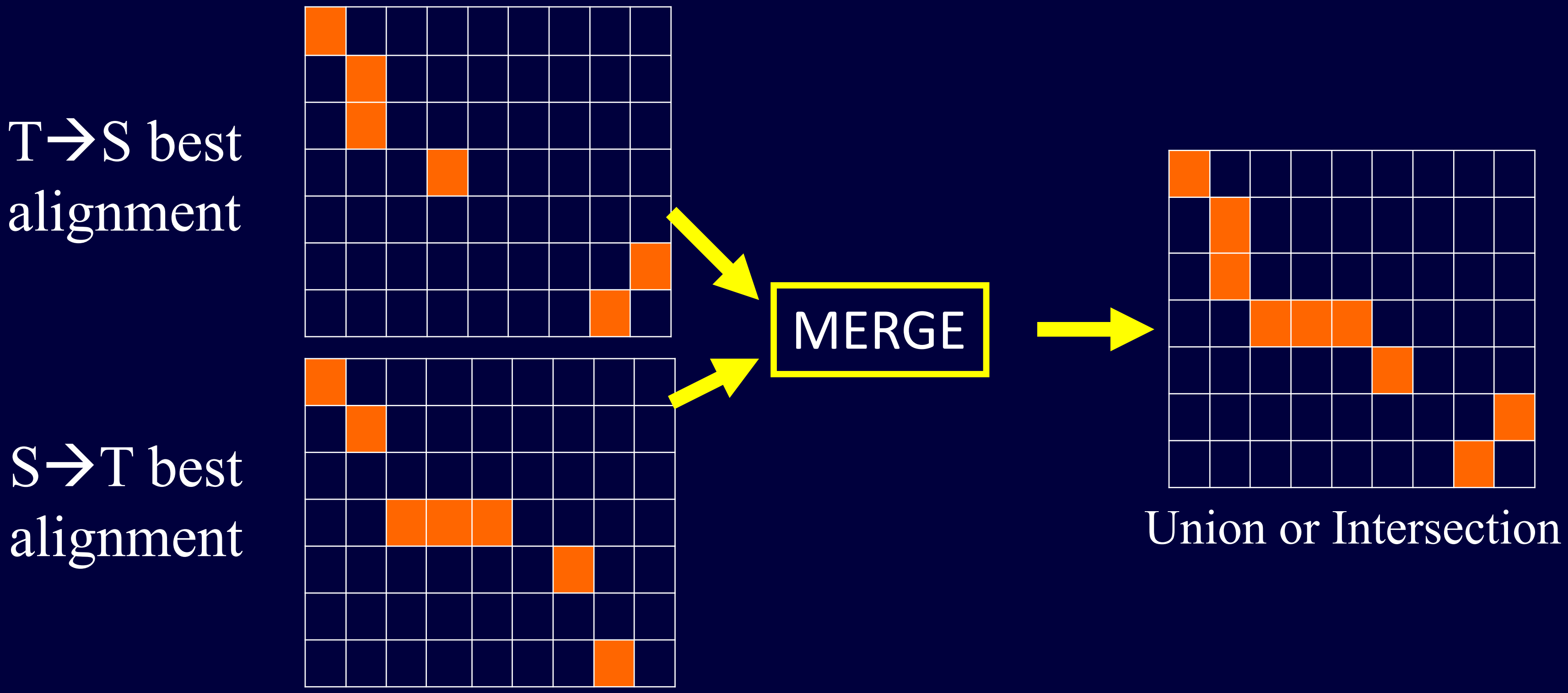
	Maria	no	dió	una	bofetada	a	la	bruja	verde
Mary	■								
did		■							
not		■							
slap			■	■	■				
the							■		
green									■
witch								■	

This word-to-word alignment is a by-product of training a translation model like IBM-Model-3.

This is the best (or “Viterbi”) alignment.

IBM Models are 1-to-Many

- Run IBM-style aligner both directions, then merge:



T→S best alignment

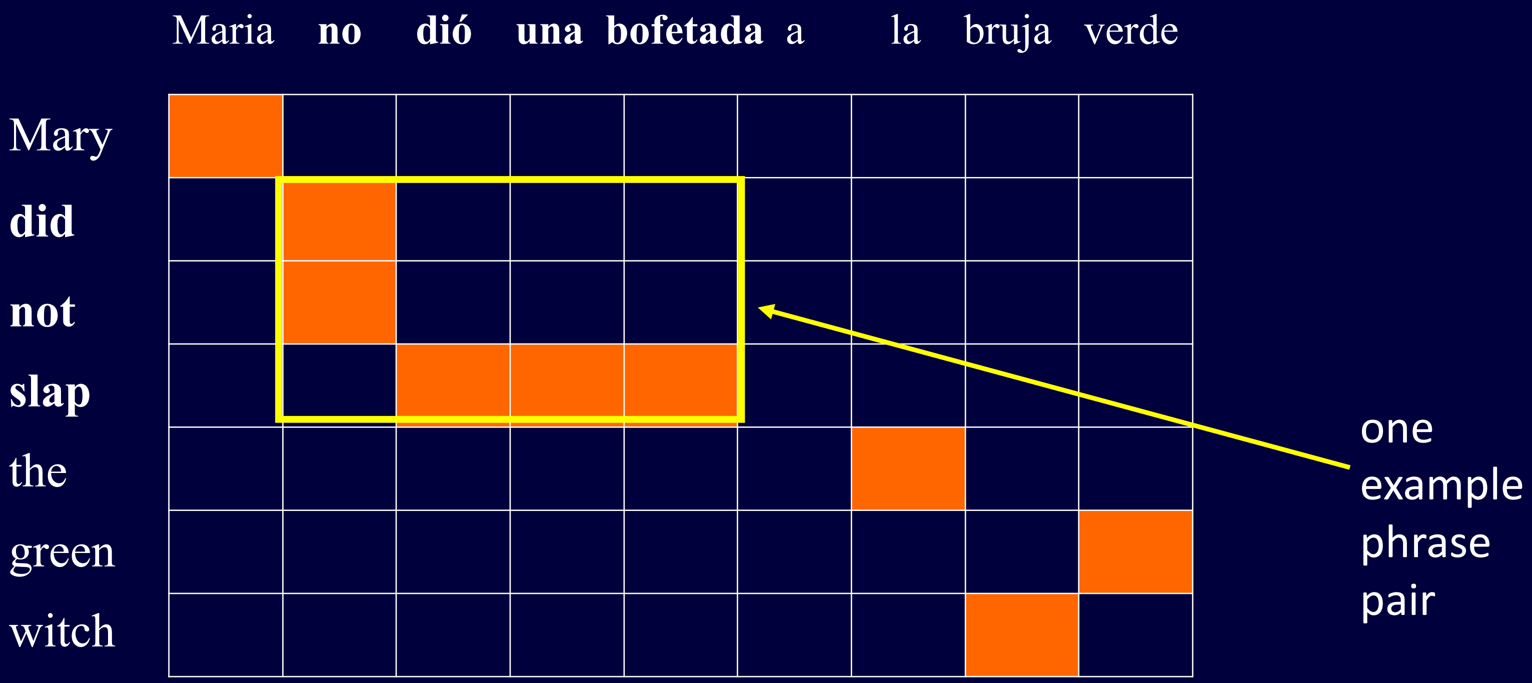
S→T best alignment

MERGE

Union or Intersection

How to Learn the Phrase Translation Table?

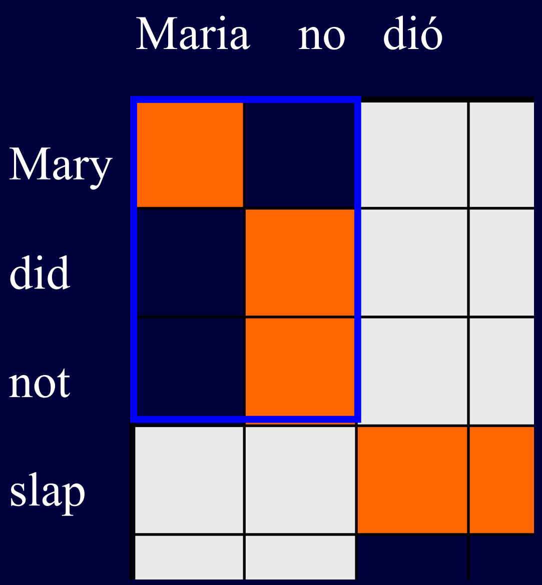
- Collect all phrase pairs *that are consistent with the word alignment*



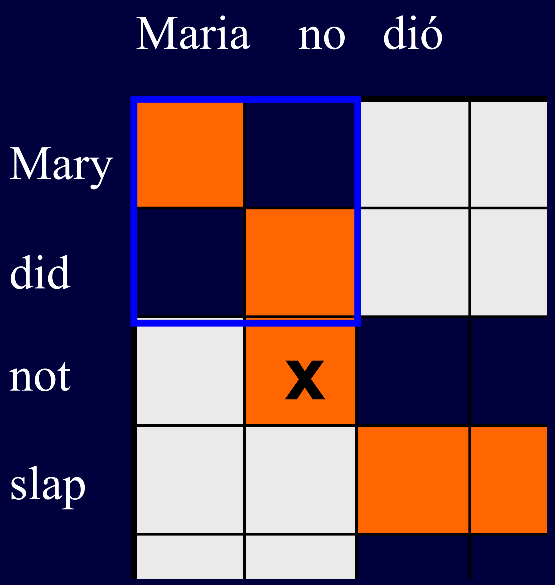
	Maria	**no**	**dió**	**una**	**bofetada**	a	la	bruja	verde
Mary	■								
did		■							
not		■							
slap			■	■	■				
the							■		
green									■
witch								■	

one example phrase pair

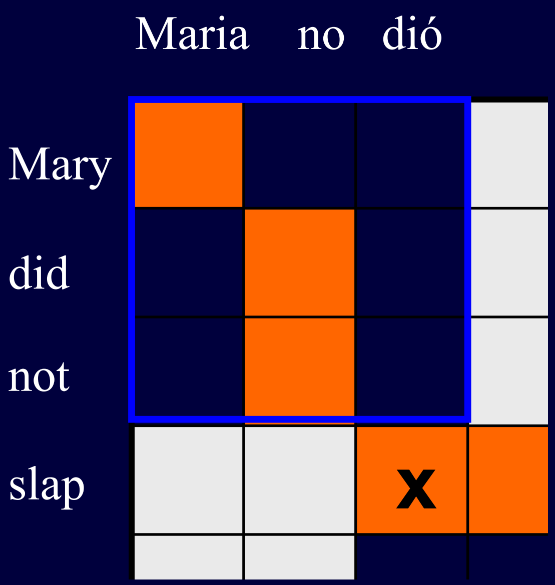
Word Alignment Consistent Phrases



consistent



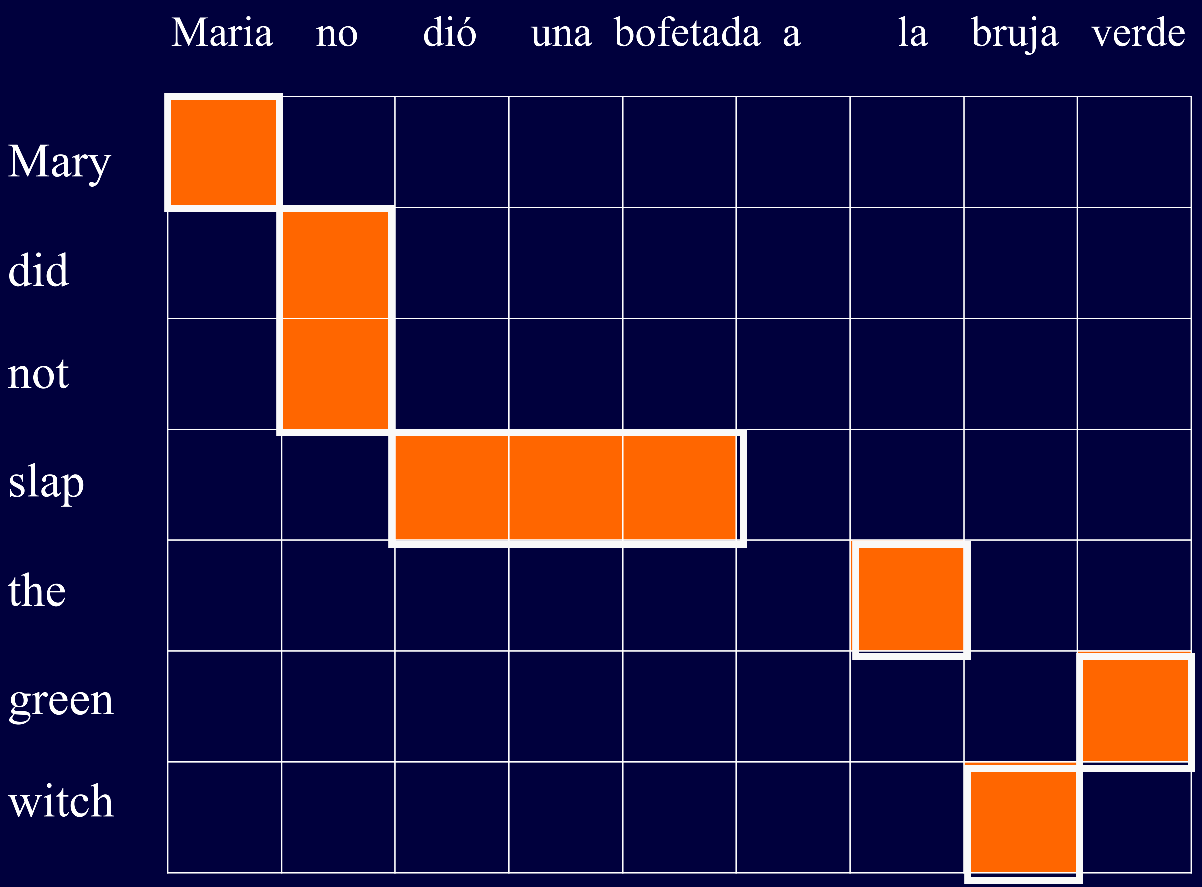
inconsistent



inconsistent

Phrase alignment must contain all alignment points for all the words in both phrases!

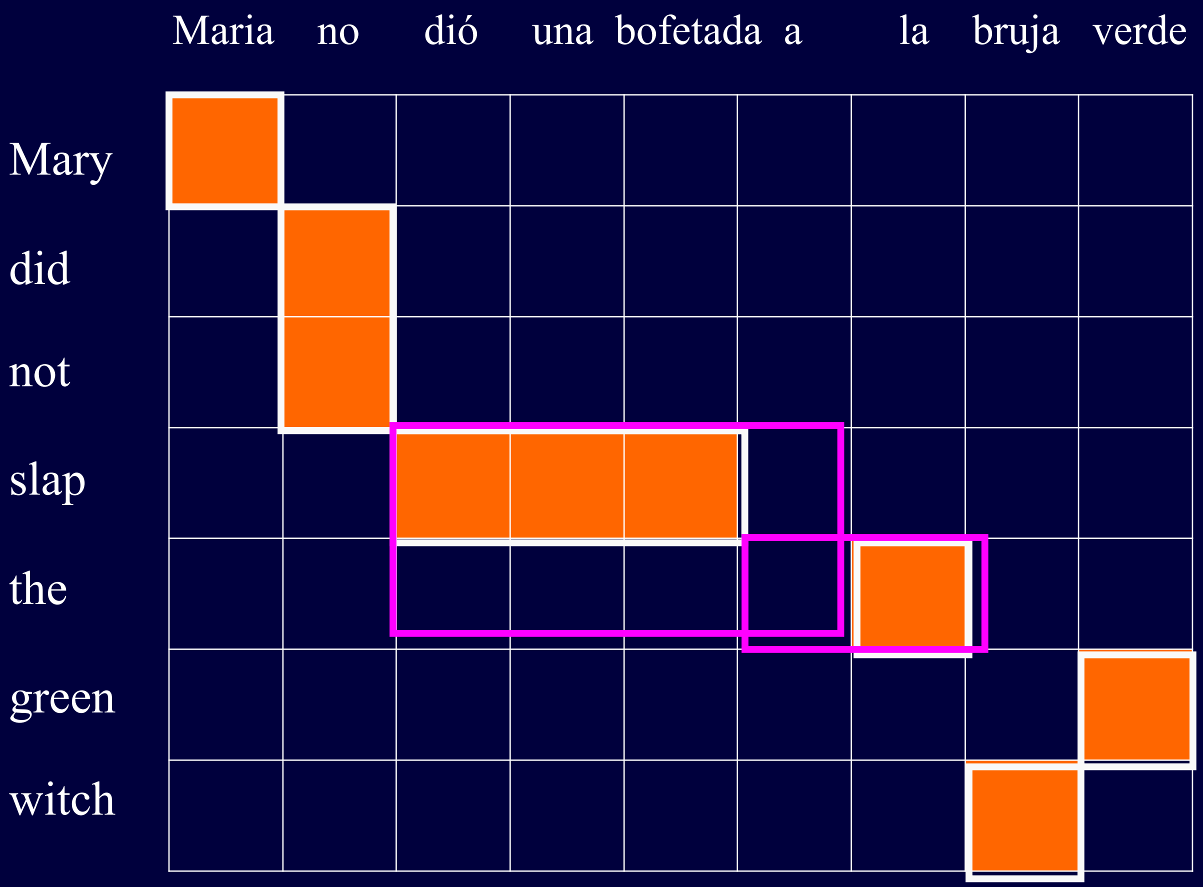
Word Alignment Induced Phrases



	Maria	no	dió	una	bofetada	a	la	bruja	verde
Mary	■								
did		■							
not		■							
slap			■	■	■				
the							■		
green									■
witch								■	

(Maria, Mary) (no, did not) (slap, dió una bofetada) (la, the) (bruja, witch) (verde, green)

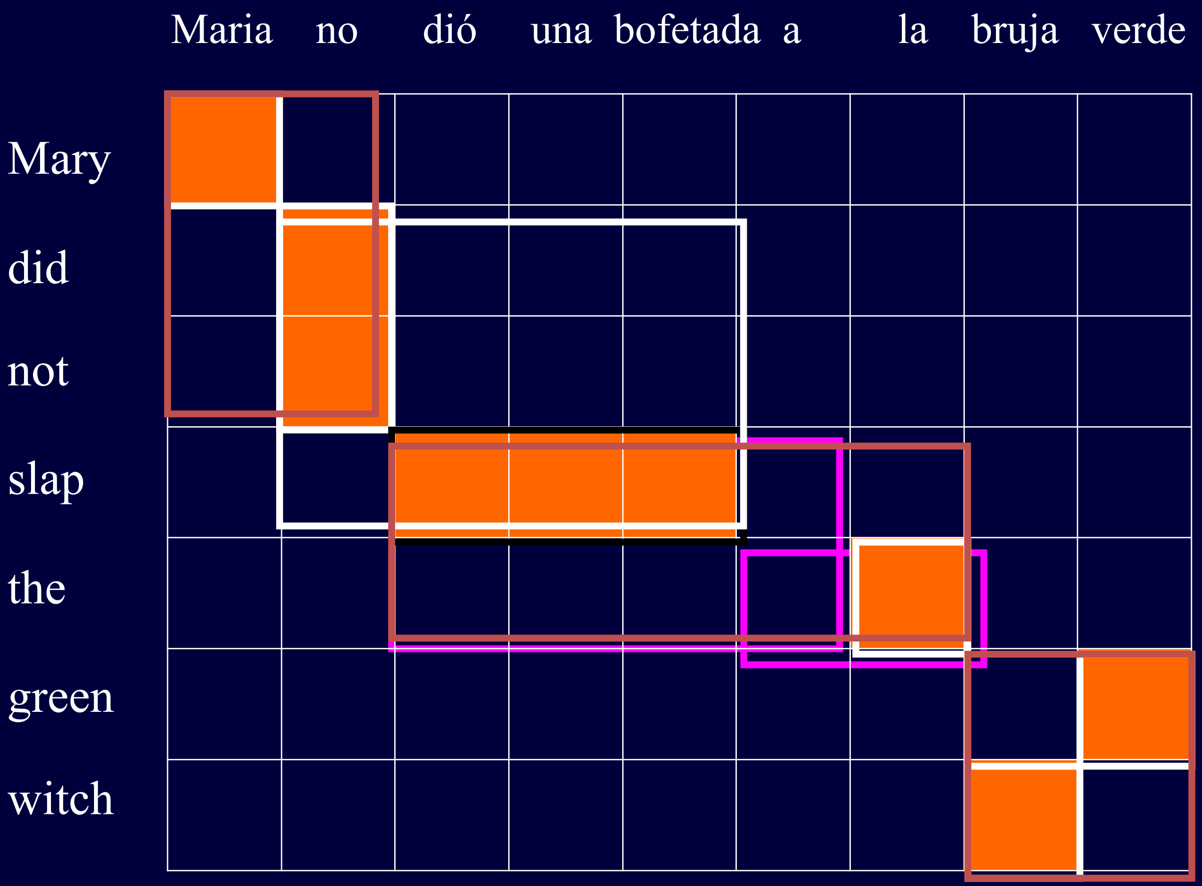
Word Alignment Induced Phrases



(Maria, Mary) (no, did not) (slap, dió una bofetada) (la, the) (bruja, witch) (verde, green)

(a la, the) (dió una bofetada a, slap the)

Word Alignment Induced Phrases



	Maria	no	dió	una	bofetada	a	la	bruja	verde
Mary	■								
did		■							
not		■							
slap			■	■	■				
the							■		
green									■
witch								■	

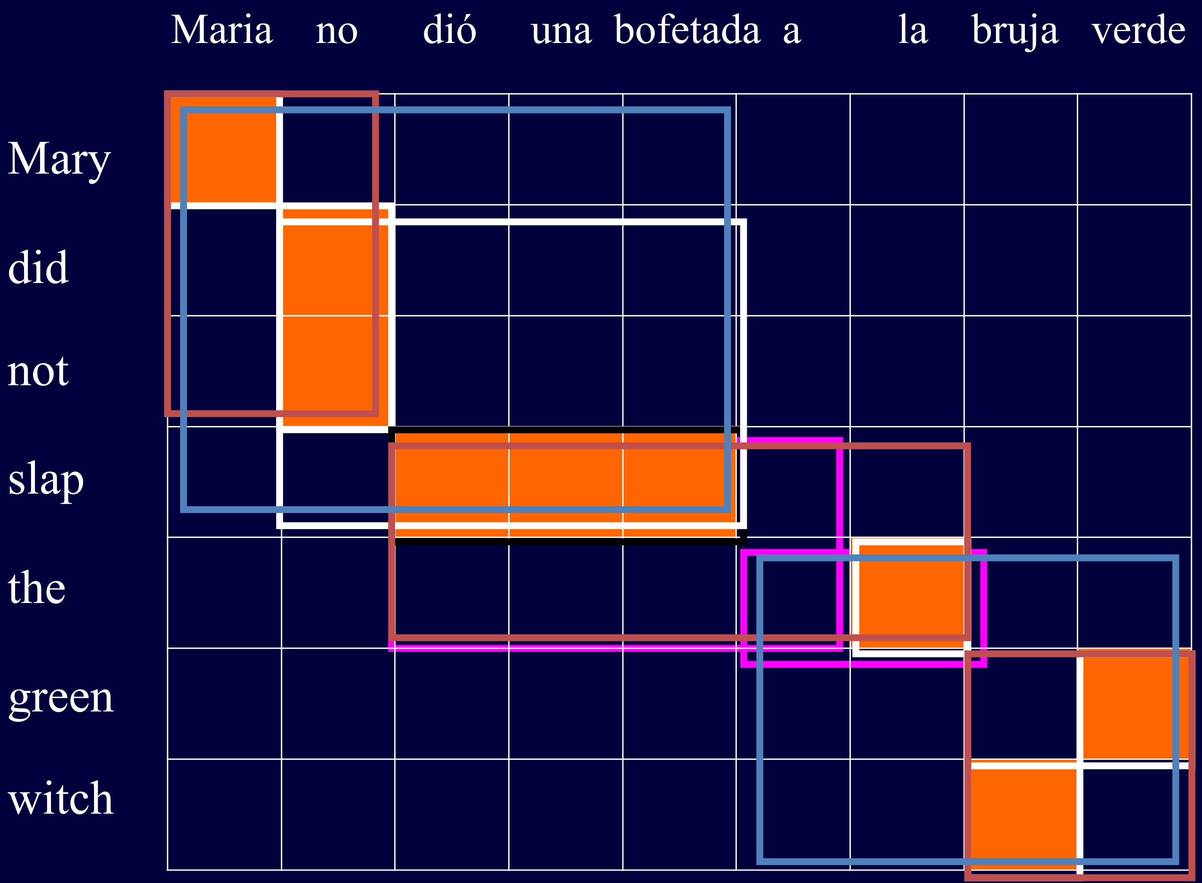
(Maria, Mary) (no, did not) (slap, dió una bofetada) (la, the) (bruja, witch) (verde, green)

(a la, the) (dió una bofetada a, slap the)

(Maria no, Mary did not) (no dió una bofetada, did not slap), (dió una bofetada a la, slap the)

(bruja verde, green witch)

Word Alignment Induced Phrases



(Maria, Mary) (no, did not) (slap, dió una bofetada) (la, the) (bruja, witch) (verde, green)

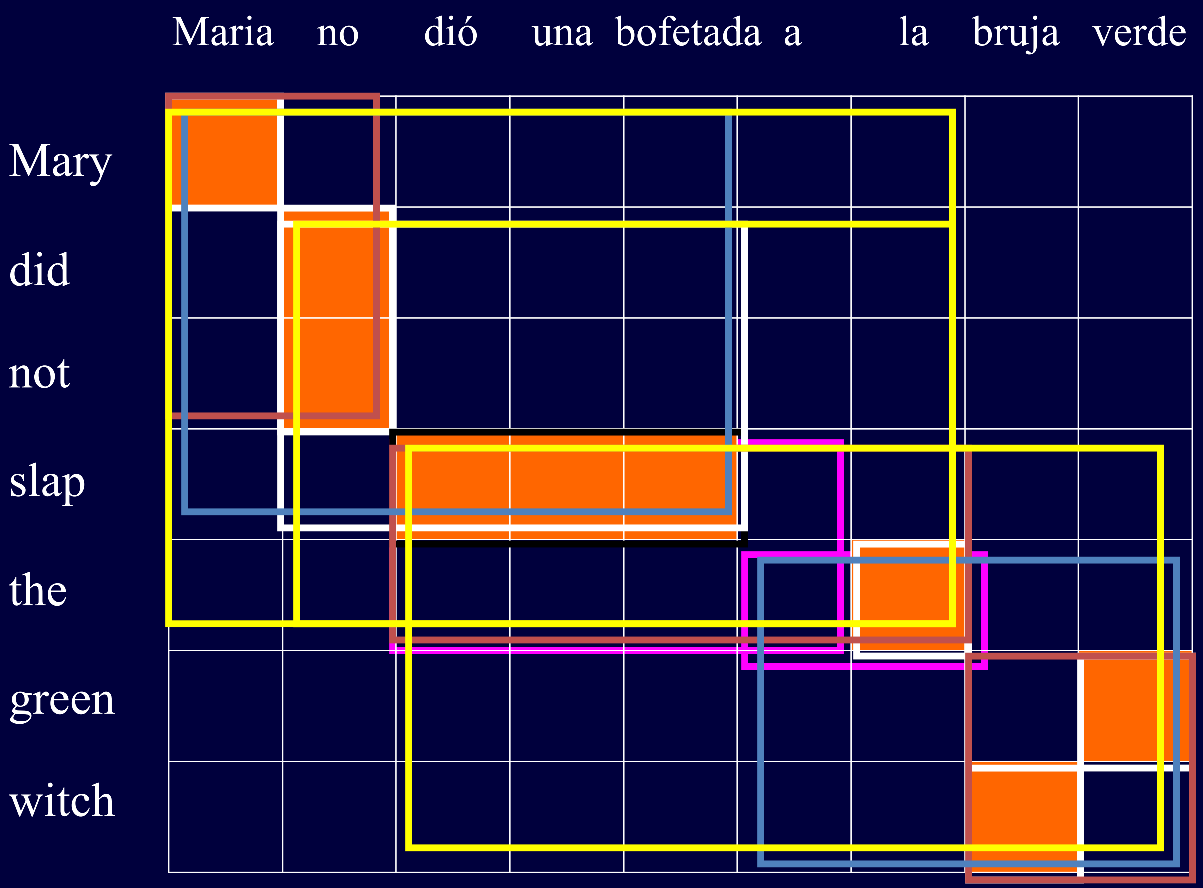
(a la, the) (dió una bofetada a, slap the)

(Maria no, Mary did not) (no dió una bofetada, did not slap), (dió una bofetada a la, slap the)

(bruja verde, green witch) (Maria no dió una bofetada, Mary did not slap)

(a la bruja verde, the green witch)

Word Alignment Induced Phrases



(Maria, Mary) (no, did not) (slap, dió una bofetada) (la, the) (bruja, witch) (verde, green)

(a la, the) (dió una bofetada a, slap the)

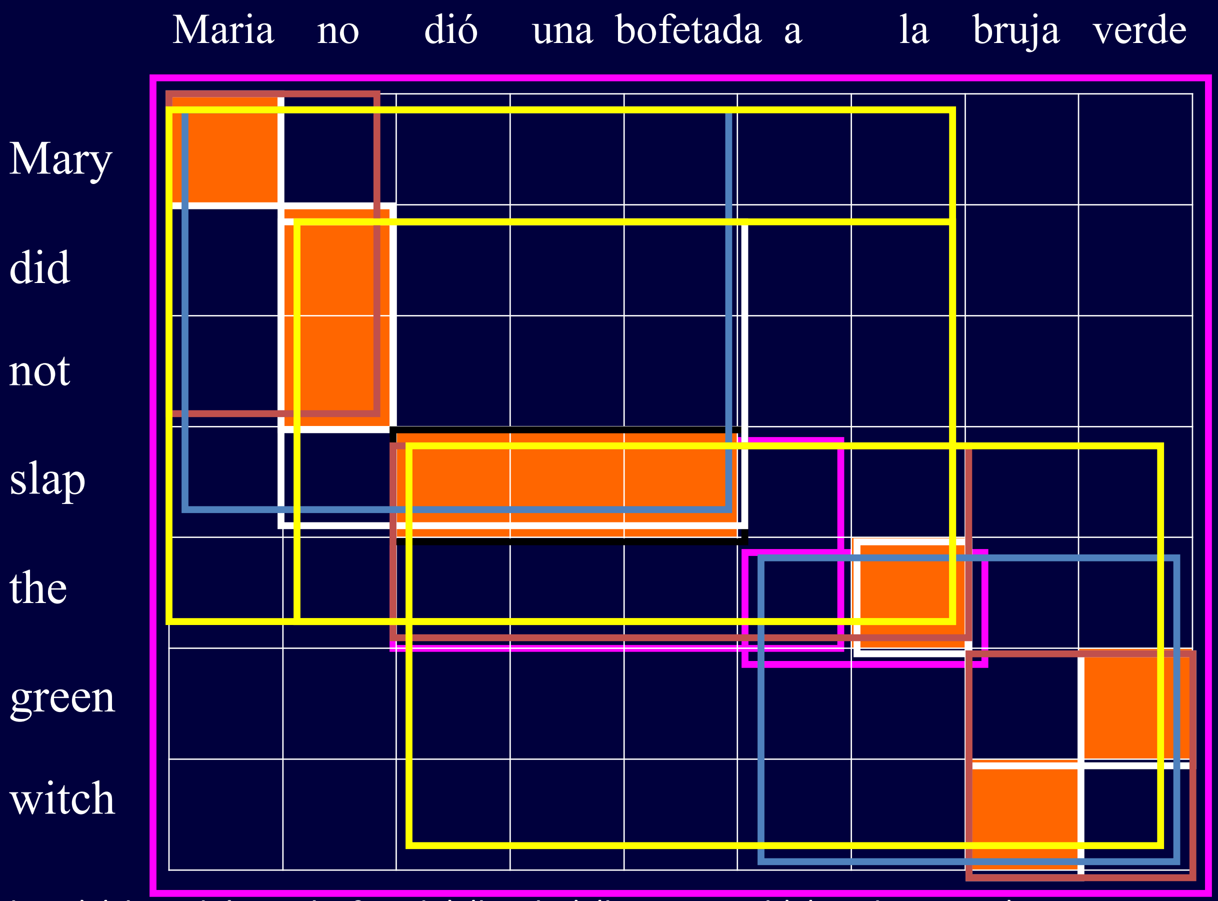
(Maria no, Mary did not) (no dió una bofetada, did not slap), (dió una bofetada a la, slap the)

(bruja verde, green witch) (Maria no dió una bofetada, Mary did not slap)

(a la bruja verde, the green witch) (Maria no dió una bofetada a la, Mary did not slap the)

(no dió una bofetada a la, did not slap the) (dió una bofetada a la bruja verde, slap the green witch)

Word Alignment Induced Phrases



(Maria, Mary) (no, did not) (slap, dió una bofetada) (la, the) (bruja, witch) (verde, green)

(a la, the) (dió una bofetada a, slap the)

(Maria no, Mary did not) (no dió una bofetada, did not slap), (dió una bofetada a la, slap the)

(bruja verde, green witch) (Maria no dió una bofetada, Mary did not slap)

(a la bruja verde, the green witch) (Maria no dió una bofetada a la, Mary did not slap the)

(no dió una bofetada a la, did not slap the) (dió una bofetada a la bruja verde, slap the green witch)

(Maria no dió una bofetada a la bruja verde, Mary did not slap the green witch)

Phrase Pair Probabilities

- A certain phrase pair (s-s-s, t-t-t) may appear many times across the bilingual corpus.
  - We hope so!
- So, now we have a vast list of phrase pairs and their frequencies – how to assign probabilities?

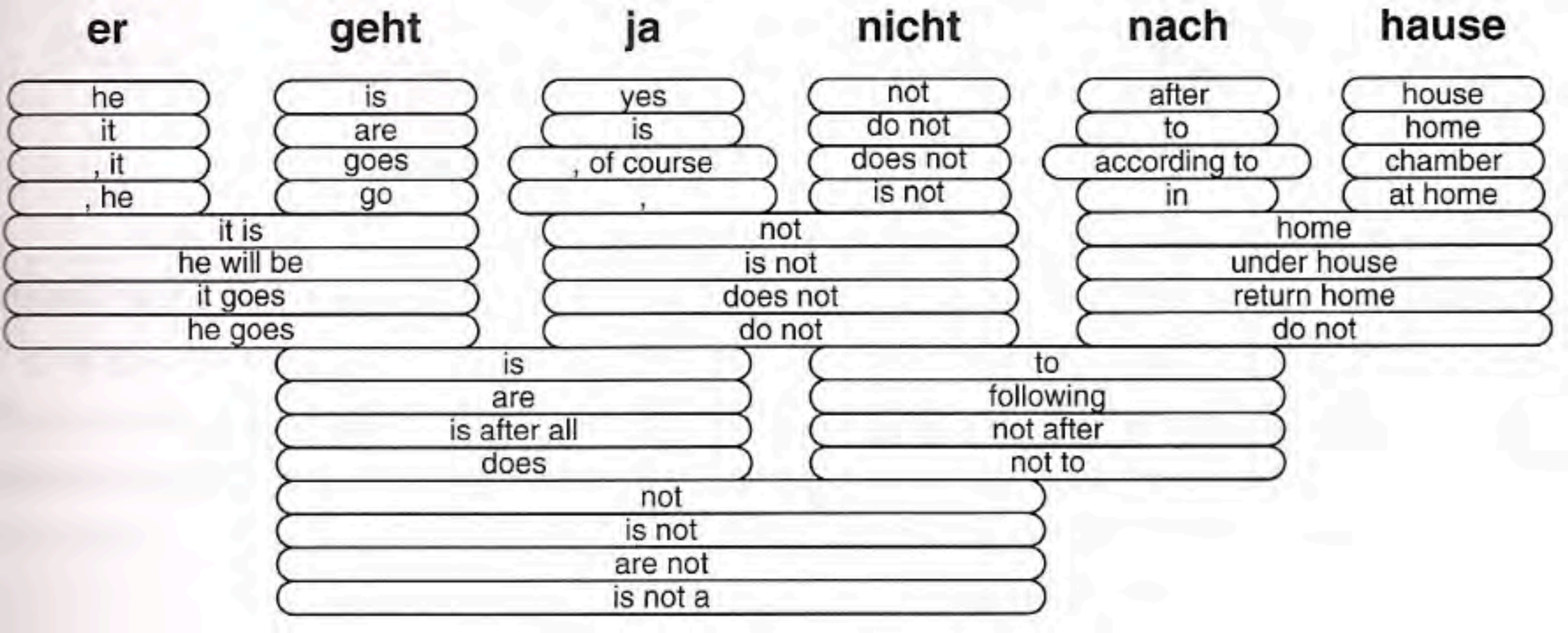
Phrase-based SMT

- After doing this to millions of sentences
  - For each phrase pair (t, s)
    - Count how many times s occurs
    - Count how many times s is translated to t
    - Estimate p(t | s)

Decoding

- During decoding
  - a sentence is segmented into “phrases” in all possible ways
  - each such phrase is then “translated” to the target phrases in all possible ways
  - Translations are also moved around
  - Resulting target sentences are scored with the target language model
- The decoder actually does NOT actually enumerate all possible translations or all possible target sentences
  - Pruning

Decoding



er geht ja nicht nach hause

er: he / it / , it / , he

geht: is / are / goes / go

ja: yes / is / , of course / ,

nicht: not / do not / does not / is not

nach: after / to / according to / in

hause: house / home / chamber / at home

er geht: it is / he will be / it goes / he goes

ja nicht: not / is not / does not / do not

nach hause: home / under house / return home / do not

geht ja: is / are / is after all / does

nicht nach: to / following / not after / not to

geht ja nicht: not / is not / are not / is not a

Basic Model, Revisited

$$\underset{t}{\operatorname{argmax}}\ P(t \mid s) \ =$$

$$\underset{t}{\operatorname{argmax}}\ P(t) \times P(s \mid t) \,/\, P(s) \ =$$

$$\underset{t}{\operatorname{argmax}}\ P(t) \times P(t \mid s)$$

Basic Model, Revisited

$$\operatorname*{argmax}_{t} P(t \mid s) =$$

$$\operatorname*{argmax}_{t} P(t) \times P(s \mid t) / P(s) =$$

$$\operatorname*{argmax}_{t} P(t)^{2.4} \times P(t \mid s)$$ seems to work better

Basic Model, Revisited

$$\underset{t}{\operatorname{argmax}}\ P(t \mid s) \ =$$

$$\underset{t}{\operatorname{argmax}}\ P(t) \times P(s \mid t) \,/\, P(s) \quad =$$

$$\underset{t}{\operatorname{argmax}}\ P(t)^{2.4} \times P(t \mid s) * \text{length}(t)^{1.1}$$

Rewards longer hypotheses, since these are unfairly punished by p(t)

Basic Model, Revisited

$$\underset{e}{\operatorname{argmax}}\ \underbrace{P(t)^{2.4} \times P(s \mid t) \times \text{length}(t)^{1.1} \times KS^{3.7} \ldots}_{}$$

Lots of **knowledge sources** vote on any given hypothesis.

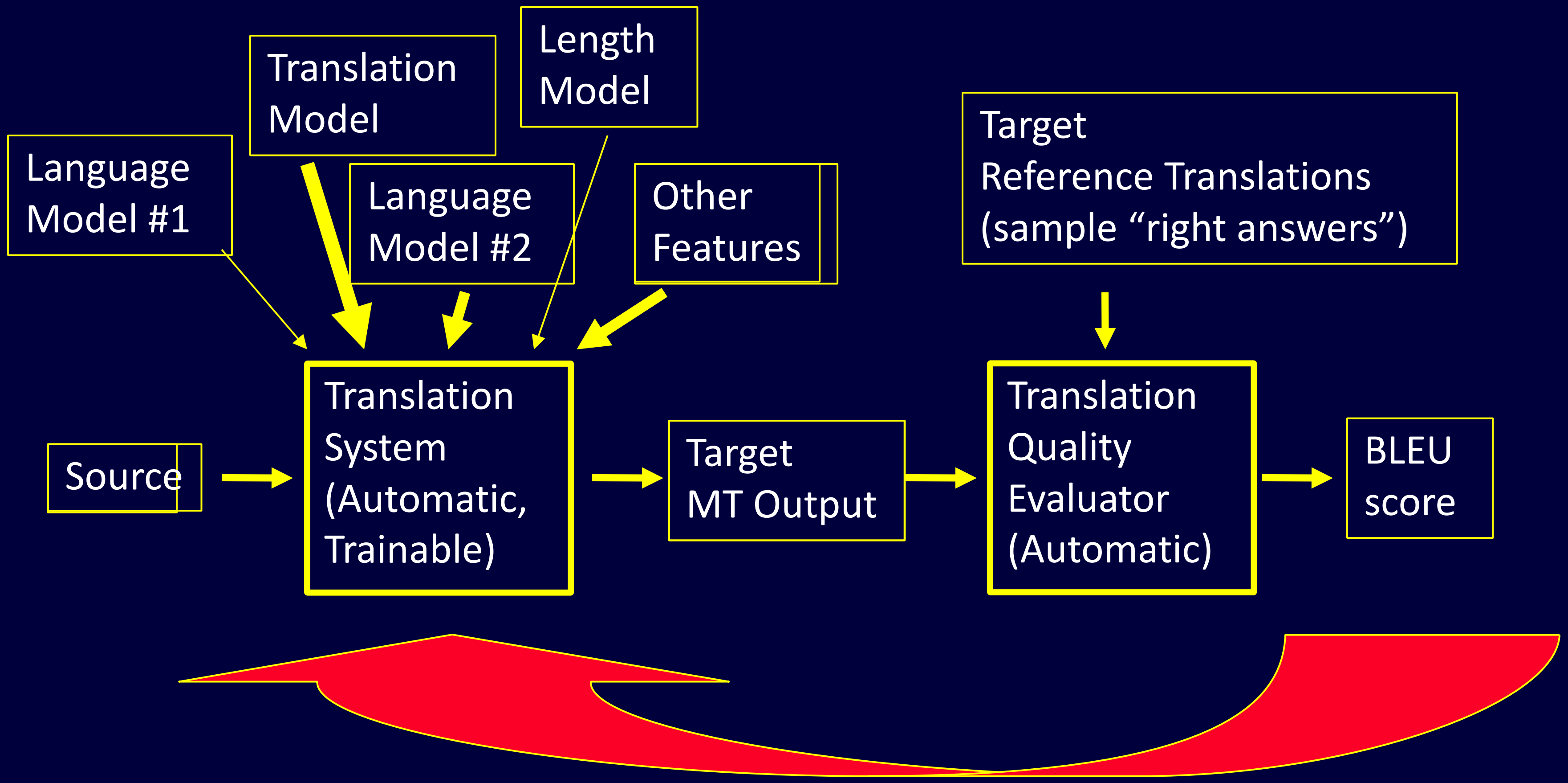
“Knowledge source” = “feature function” = “score component”.

Feature function simply scores a hypothesis with a real value.

(May be binary, as in “e has a verb”).

Problem: How to set the exponent weights?

Maximum BLEU Training

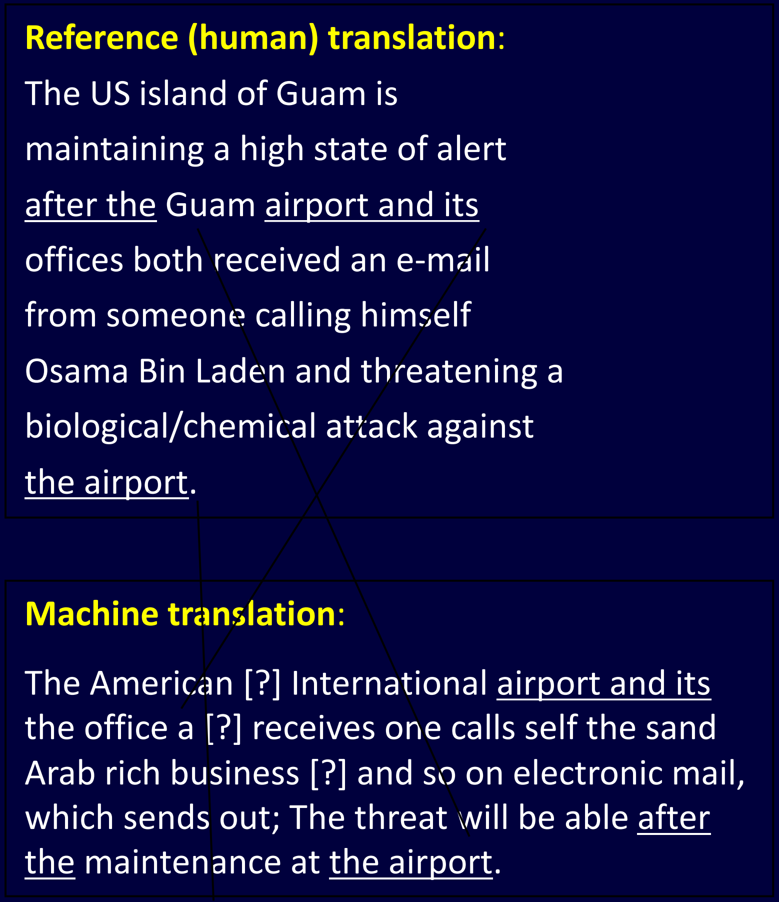

Language Model #1
Translation Model
Language Model #2
Length Model
Other Features
Target Reference Translations (sample “right answers”)
Source
Translation System (Automatic, Trainable)
Target MT Output
Translation Quality Evaluator (Automatic)
BLEU score

Learning Algorithm for Directly Reducing Translation Error
Yields big improvements in quality.

Automatic Machine Translation Evaluation

- Objective
- Inspired by the Word Error Rate metric used by ASR research
- Measuring the “closeness” between the MT hypothesis and human reference translations
  - Precision: n-gram precision
  - Recall:
    - Against the best matched reference
    - Approximated by brevity penalty
- Cheap, fast
- Highly correlated with human evaluations
- MT research has greatly benefited from automatic evaluations
- Typical metrics: BLEU, NIST, F-Score, Meteor, TER

BLEU Evaluation



Reference (human) translation:

The US island of Guam is maintaining a high state of alert after the Guam airport and its offices both received an e-mail from someone calling himself Osama Bin Laden and threatening a biological/chemical attack against the airport.

Machine translation:

The American [?] International airport and its the office a [?] receives one calls self the sand Arab rich business [?] and so on electronic mail, which sends out; The threat will be able after the maintenance at the airport.

N-gram precision (score between 0 & 1)

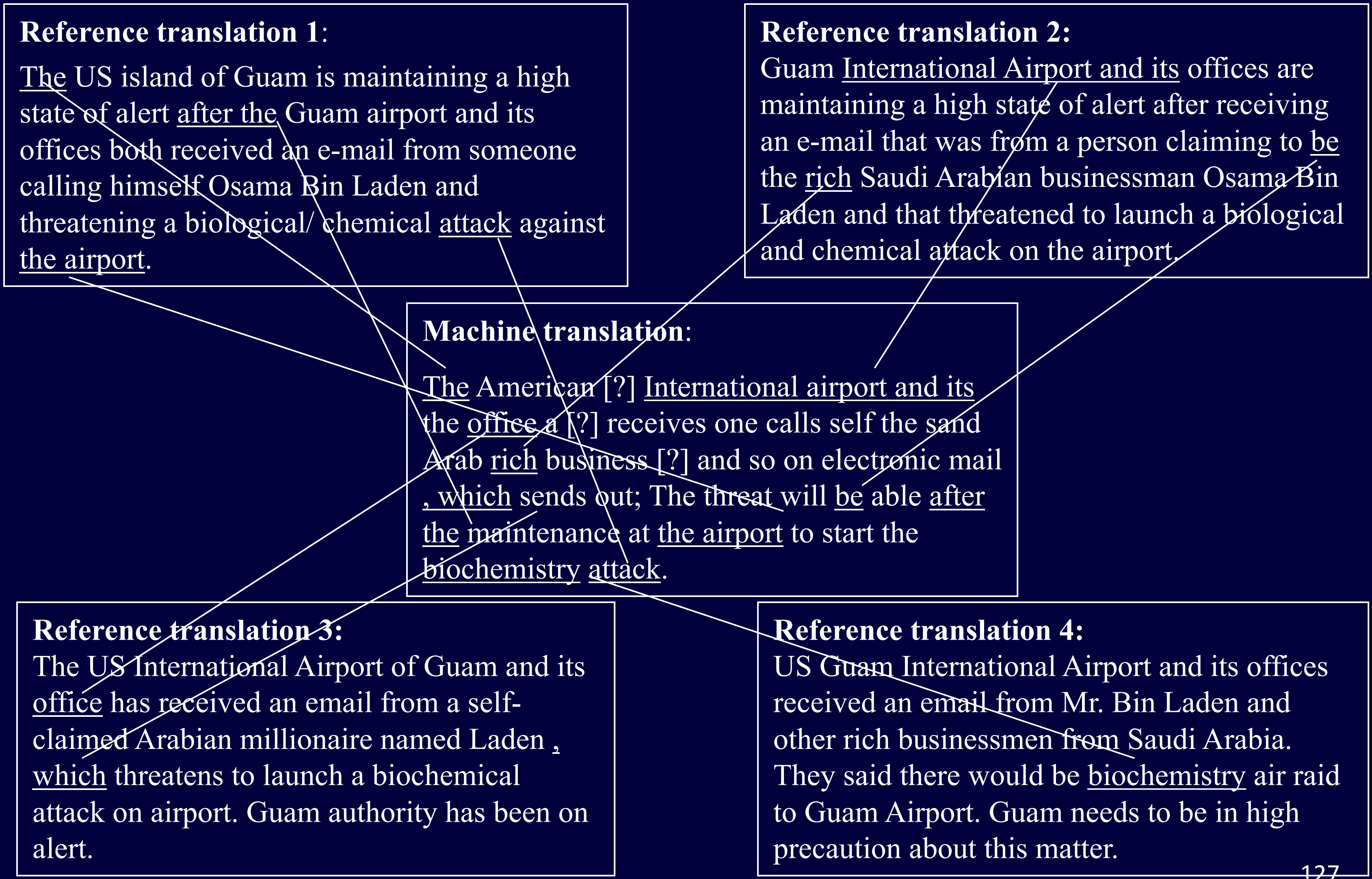
- what % of machine n-grams (a sequence of words) can be found in the reference translation?

Brevity Penalty

- Can’t just type out single word “the’’ (precision 1.0!)

Extremely hard to trick the system,
i.e. find a way to change MT output so that BLEU score increases, but quality doesn’t.

More Reference Translations are Better



Reference translation 1:
The US island of Guam is maintaining a high state of alert after the Guam airport and its offices both received an e-mail from someone calling himself Osama Bin Laden and threatening a biological/ chemical attack against the airport.

Reference translation 2:
Guam International Airport and its offices are maintaining a high state of alert after receiving an e-mail that was from a person claiming to be the rich Saudi Arabian businessman Osama Bin Laden and that threatened to launch a biological and chemical attack on the airport.

Machine translation:
The American [?] International airport and its the office a [?] receives one calls self the sand Arab rich business [?] and so on electronic mail , which sends out; The threat will be able after the maintenance at the airport to start the biochemistry attack.

Reference translation 3:
The US International Airport of Guam and its office has received an email from a self-claimed Arabian millionaire named Laden , which threatens to launch a biochemical attack on airport. Guam authority has been on alert.

Reference translation 4:
US Guam International Airport and its offices received an email from Mr. Bin Laden and other rich businessmen from Saudi Arabia. They said there would be biochemistry air raid to Guam Airport. Guam needs to be in high precaution about this matter.

BLEU in Action

- Reference Translation: *The gunman was shot to death by the police .*

- The gunman was shot kill .
- Wounded police jaya of
- The gunman was shot dead by the police .
- The gunman arrested by police kill .
- The gunmen were killed .
- The gunman was shot to death by the police .
- The ringer is killed by the police .
- Police killed the gunman .

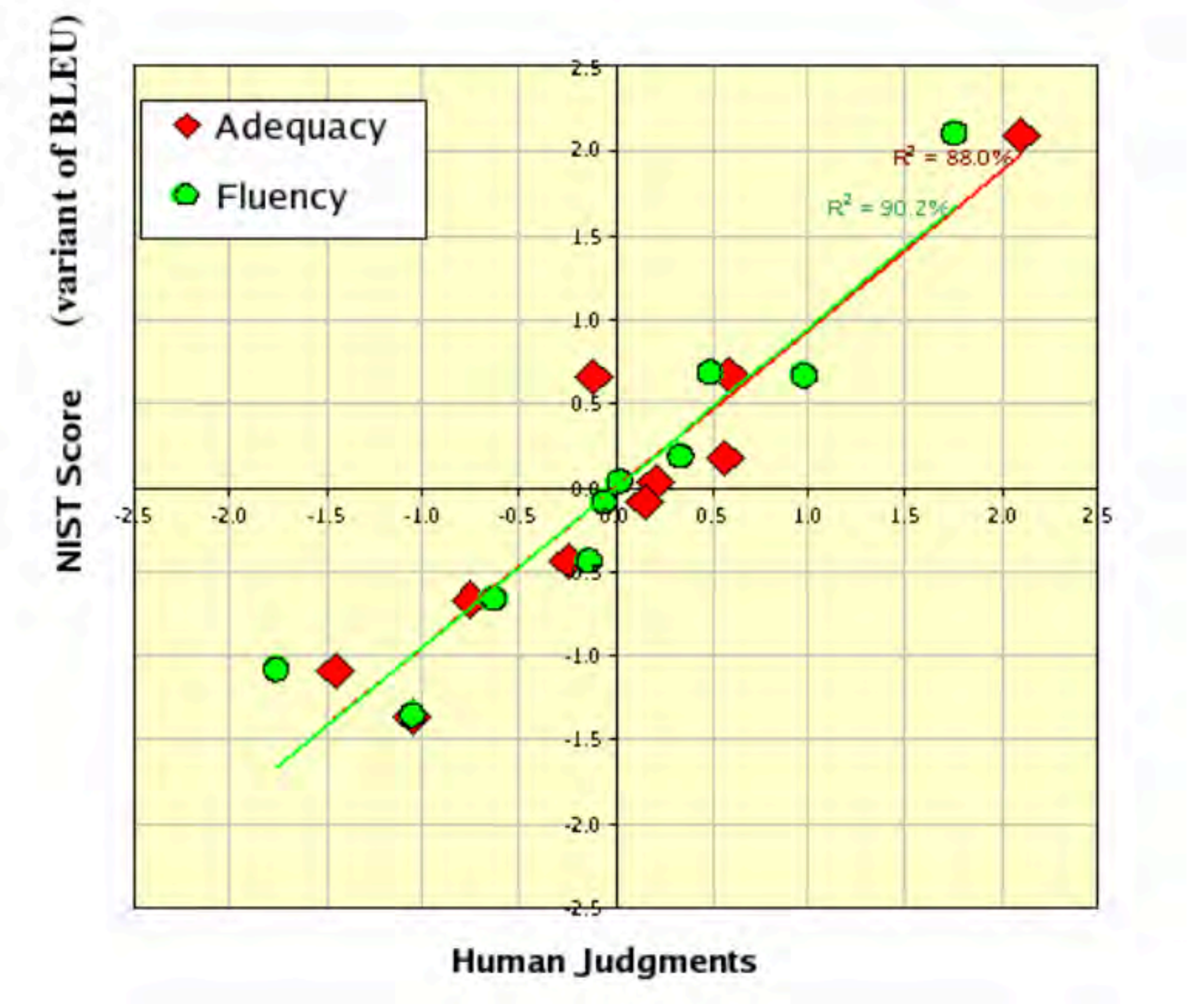
- Green = 4-gram match (good!)   Red = unmatched word (bad!)

BLEU Formulation

$$BLEU = \min(1, \frac{output - length}{reference - length})\left(\prod_{i=1}^{4} precision_i\right)^{\frac{1}{4}}$$

precision$_i$: i-gram precision over the whole corpus

Correlation with Human Judgment



What About Morphology?

- Issue for handling morphologically complex languages like Turkish, Hungarian, Finnish, Arabic, etc.
  - A word contains much more information than just the root word
    - Arabic: wsyktbunha (wa+sa+ya+ktub+ūn+ha "and they will write her")
      - What are the alignments?
    - Turkish: gelebilecekmissin (gel+ebil+ecek+mis+sin (I heard) you would be coming))
      - What are the alignments?

Morphology & SMT

- Finlandiyalılaştıramadıklarımızdanmışsınızcasına
- Finlandiya+lı+laş+tır+ama+dık+lar+ımız+dan+mış+sınız+casına
- (behaving) as if you have been one of those whom we could not convert into a Finn(ish citizen)/someone from Finland

Morphology & SMT

- yapabileceksek
  - yap+abil+ecek+se+k
  - if we will be able to do (something)
- yaptırtabildiğimizde
  - yap+tır+t+tığ+ımız+da
  - when/at the time we had (someone) have (someone else) do (something)
- görüntülenebilir
  - görüntüle+n+ebil+ir
  - it can be visualize+d
- sakarlıklarından
  - sakar+lık+ları+ndan
  - of/from/due-to their clumsi+ness

Most of the time, the morpheme order is "reverse" of the corresponding English word order

Morphology and Alignment

- Remember the alignment needs to count co-occuring words
  - If one side of the parallel text has little morphology (e.g. English)
  - The other side has lots of morphology
- Lots of words on the English side either don't align or align randomly

Morphology & SMT

- If we ignore morphology
  - Large vocabulary size on the Turkish side
  - Potentially noisy alignments
  - The link activity-faaliyet is very “loose”

Word Form	**Count**	**Gloss**
faaliyet	3	activity
faaliyete	1	to the activity
faaliyetinde	1	in its activity
faaliyetler	3	activities
faaliyetlere	6	to the activities
faaliyetleri	7	their activities
faaliyetlerin	7	of the activities
faaliyetlerinde	1	in their activities
faaliyetlerine	5	to their activities
faaliyetlerini	1	their activities (accusative)
faaliyetlerinin	2	of their activities
faaliyetleriyle	1	with their activities
faaliyette	2	in (the) activity
faaliyetteki	1	that is in activity
TOTAL	**41**	

An Example E – T Translation

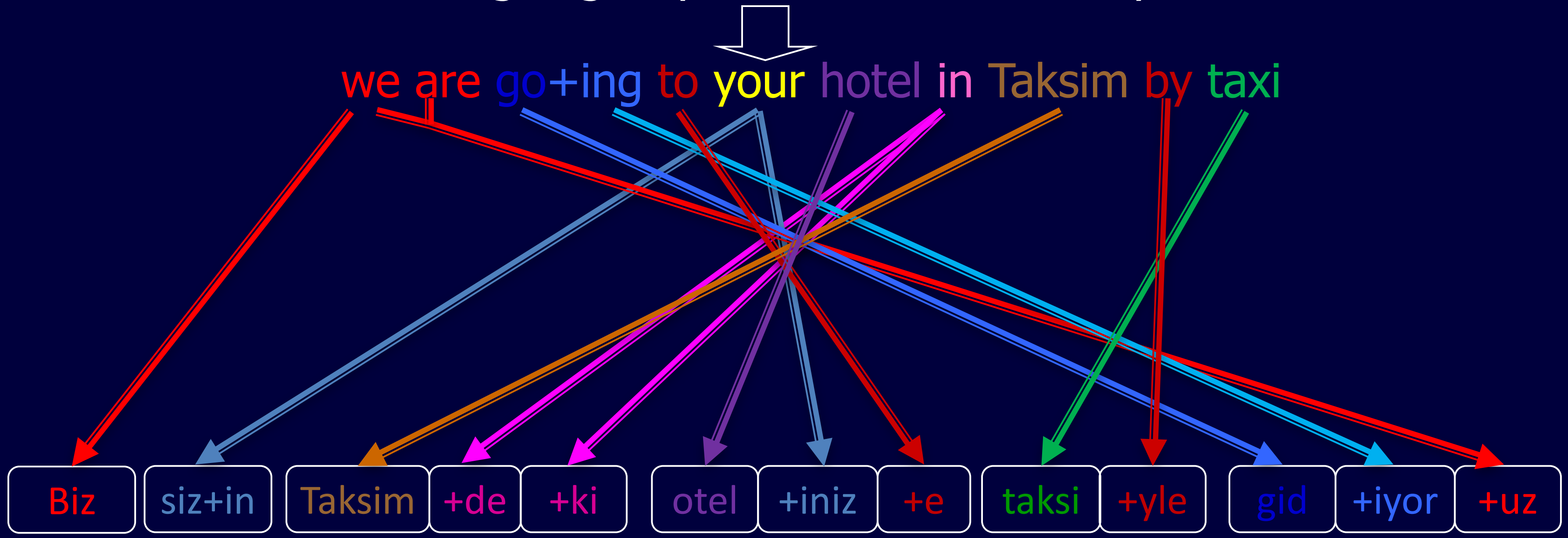
we are going to your hotel in Taksim by taxi

⇩

we are go+ing to your hotel in Taksim by taxi

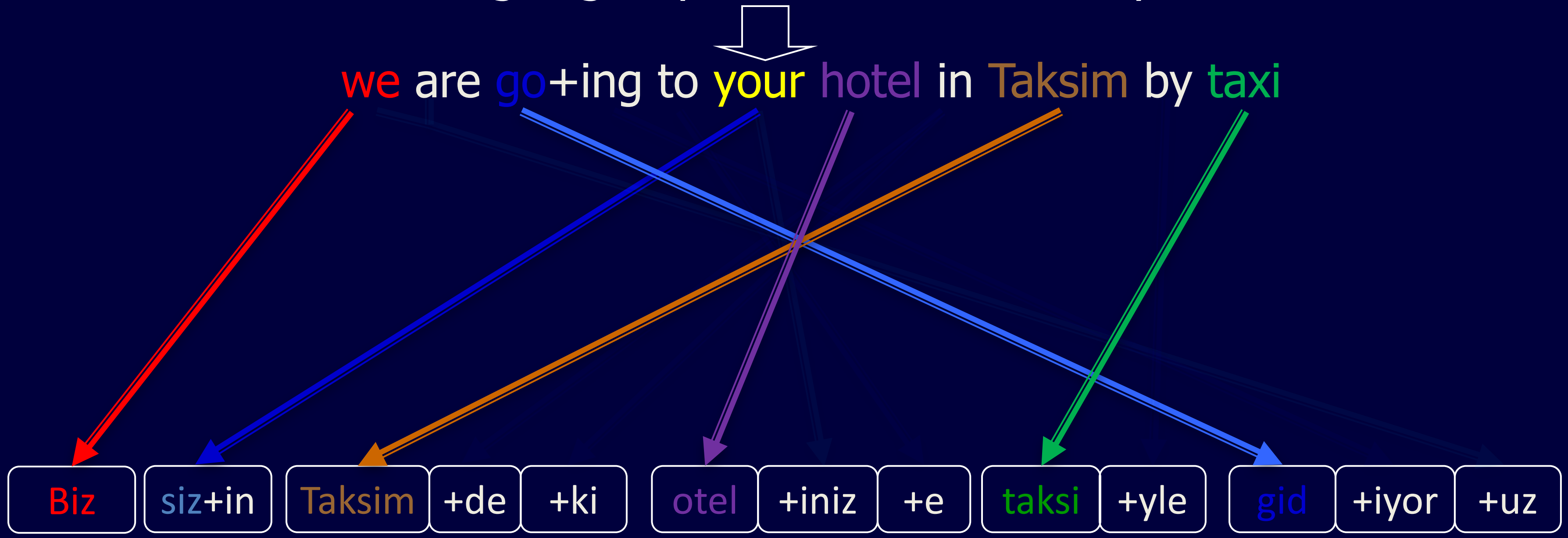
An Example E – T Translation

we are going to your hotel in Taksim by taxi



An Example E – T Translation

we are going to your hotel in Taksim by taxi

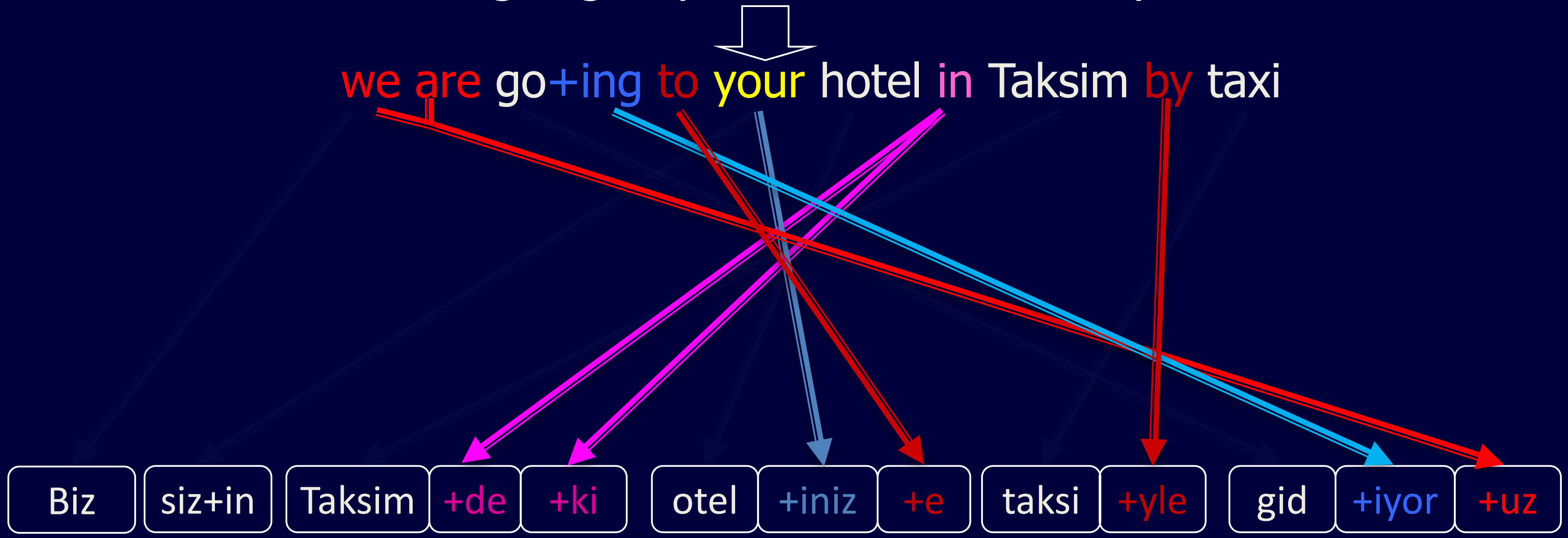


we are go+ing to your hotel in Taksim by taxi

Biz | siz+in | Taksim +de +ki | otel +iniz +e | taksi +yle | gid +iyor +uz

An Example E – T Translation

we are going to your hotel in Taksim by taxi

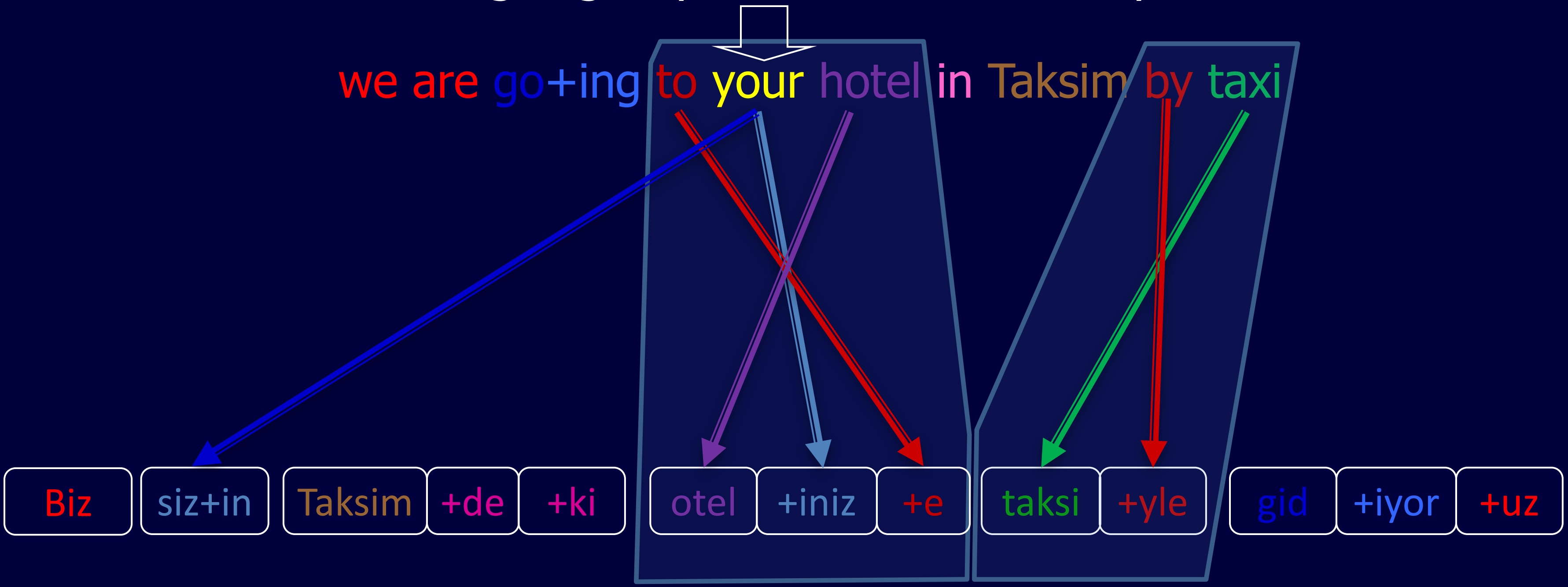


we are go+ing to your hotel in Taksim by taxi

Biz siz+in Taksim +de +ki otel +iniz +e taksi +yle gid +iyor +uz

An Example E – T Translation

we are going to your hotel in Taksim by taxi



Morphology and Parallel Texts

- Use
  - – Morphological analyzers (HLT Workshop 2)
  - – Tagger/Disambiguators (HLT Workshop 3)
- to split both sides of the parallel corpus into moprhemes

Morphology and Parallel Texts

- A typical sentence pair in this corpus looks like the following:
- Turkish:
  - kat +hl +ma ortaklık +sh +nhn uygula +hn +ma +sh , ortaklık anlaşma +sh çerçeve +sh +nda izle +hn +yacak +dhr .
- English:
  - the implementation of the accession partnership will be monitor +ed in the framework of the association agreement

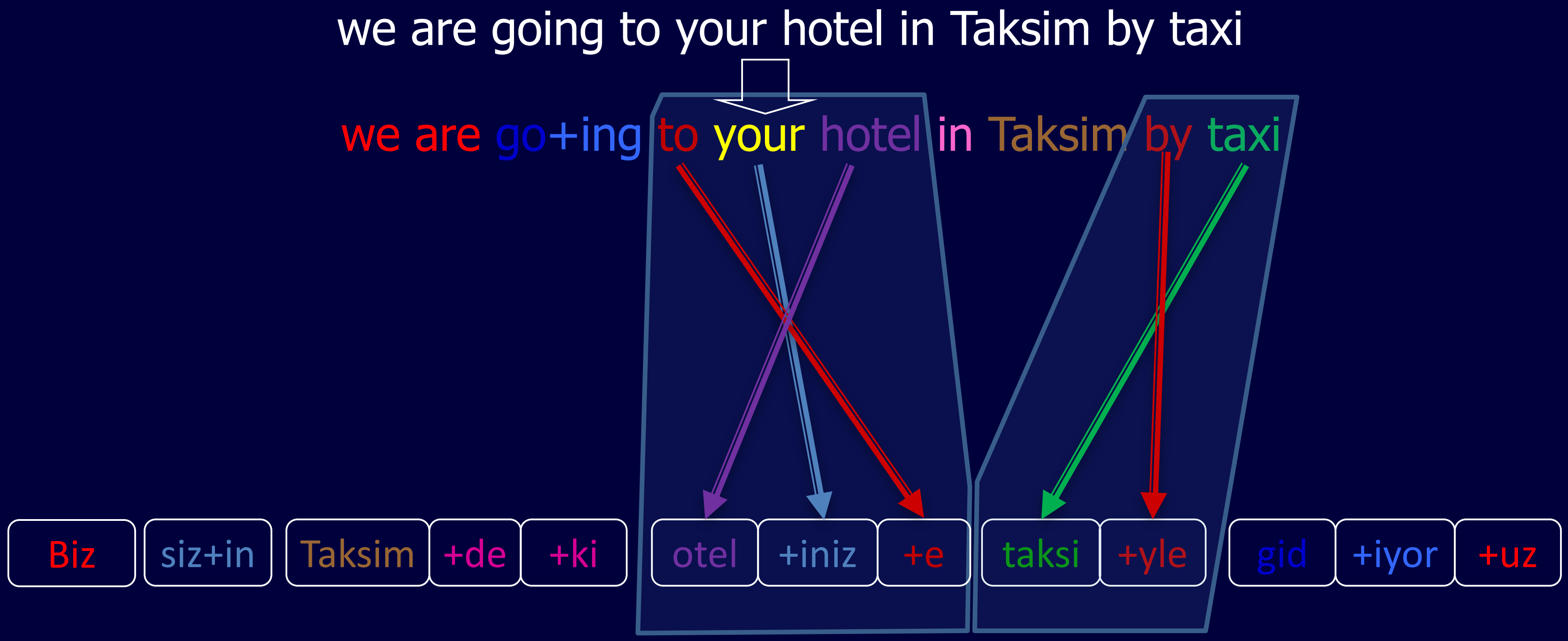
Results

- Using morphology in Phrase-based SMT certainly improves results compared to just using words
- But
  - Sentences get much longer and this hurts alignment
  - We now have an additional problem: getting the morpheme order on each word right

Syntax and Morphology Interaction

- A completely different approach
  - Instead of dividing up Turkish side into morpheme
  - Collect “stuff” on the English side to make-up “words”.
  - What is the motivation?

Syntax and Morphology Interaction


we are going to your hotel in Taksim by taxi
we are go+ing to your hotel in Taksim by taxi
Biz
siz+in
Taksim
+de
+ki
otel
+iniz
+e
taksi
+yle
gid
+iyor
+uz

Suppose we can do some **syntactic analysis on the English side**

Syntax and Morphology Interaction

we are go+ing to your hotel in Taksim by taxi

- to your hotel
  - to is the preposition related to hotel
  - your is the possessor of hotel
- to your hotel => hotel +your+to

  otel +iniz+e
  - separate content from local syntax

Syntax and Morphology Interaction

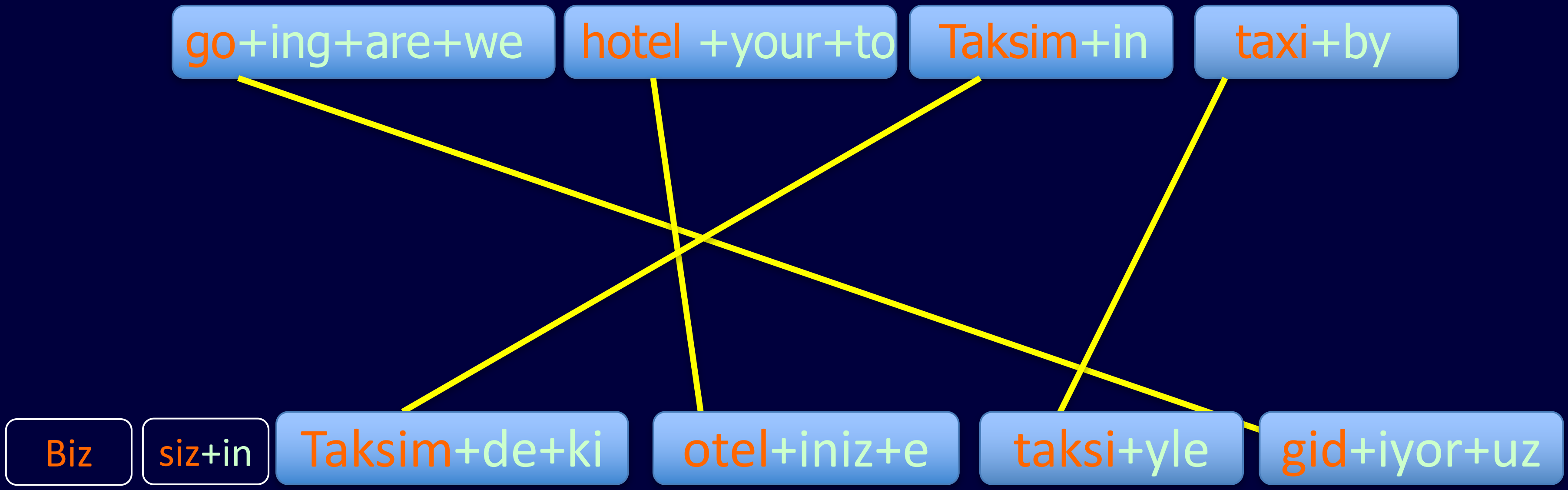
we are go+ing to your hotel in Taksim by taxi

- we are go+ing
  - – we is the subject of go
  - – are is the auxiliary of go
  - – ing is the present tense marker for go
- we are go+ing => go +ing+are+we

  gid +iyor+uz
  - – separate content from local syntax

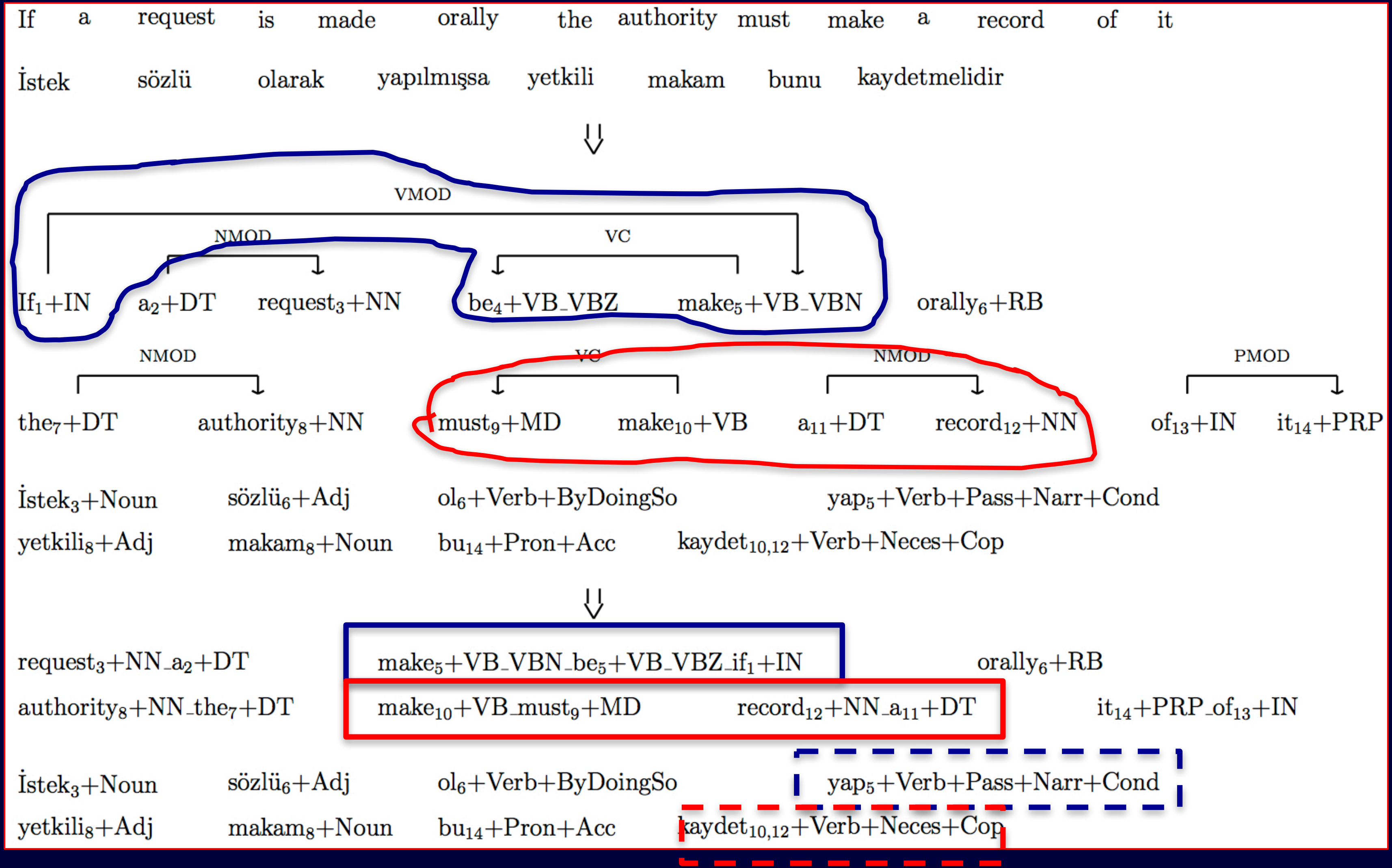
Syntax and Morphology Interaction

we are go+ing to your hotel in Taksim by taxi



Now align only based on root words – the syntax alignments just follow that

Syntax and Morphology Interaction



If a request is made orally the authority must make a record of it

İstek sözlü olarak yapılmışsa yetkili makam bunu kaydetmelidir

⇓

If$_1$+IN a$_2$+DT request$_3$+NN be$_4$+VB_VBZ make$_5$+VB_VBN orally$_6$+RB

the$_7$+DT authority$_8$+NN must$_9$+MD make$_{10}$+VB a$_{11}$+DT record$_{12}$+NN of$_{13}$+IN it$_{14}$+PRP

İstek$_3$+Noun sözlü$_6$+Adj ol$_6$+Verb+ByDoingSo yap$_5$+Verb+Pass+Narr+Cond

yetkili$_8$+Adj makam$_8$+Noun bu$_{14}$+Pron+Acc kaydet$_{10,12}$+Verb+Neces+Cop

⇓

request$_3$+NN_a$_2$+DT make$_5$+VB_VBN_be$_5$+VB_VBZ_if$_1$+IN orally$_6$+RB

authority$_8$+NN_the$_7$+DT make$_{10}$+VB_must$_9$+MD record$_{12}$+NN_a$_{11}$+DT it$_{14}$+PRP_of$_{13}$+IN

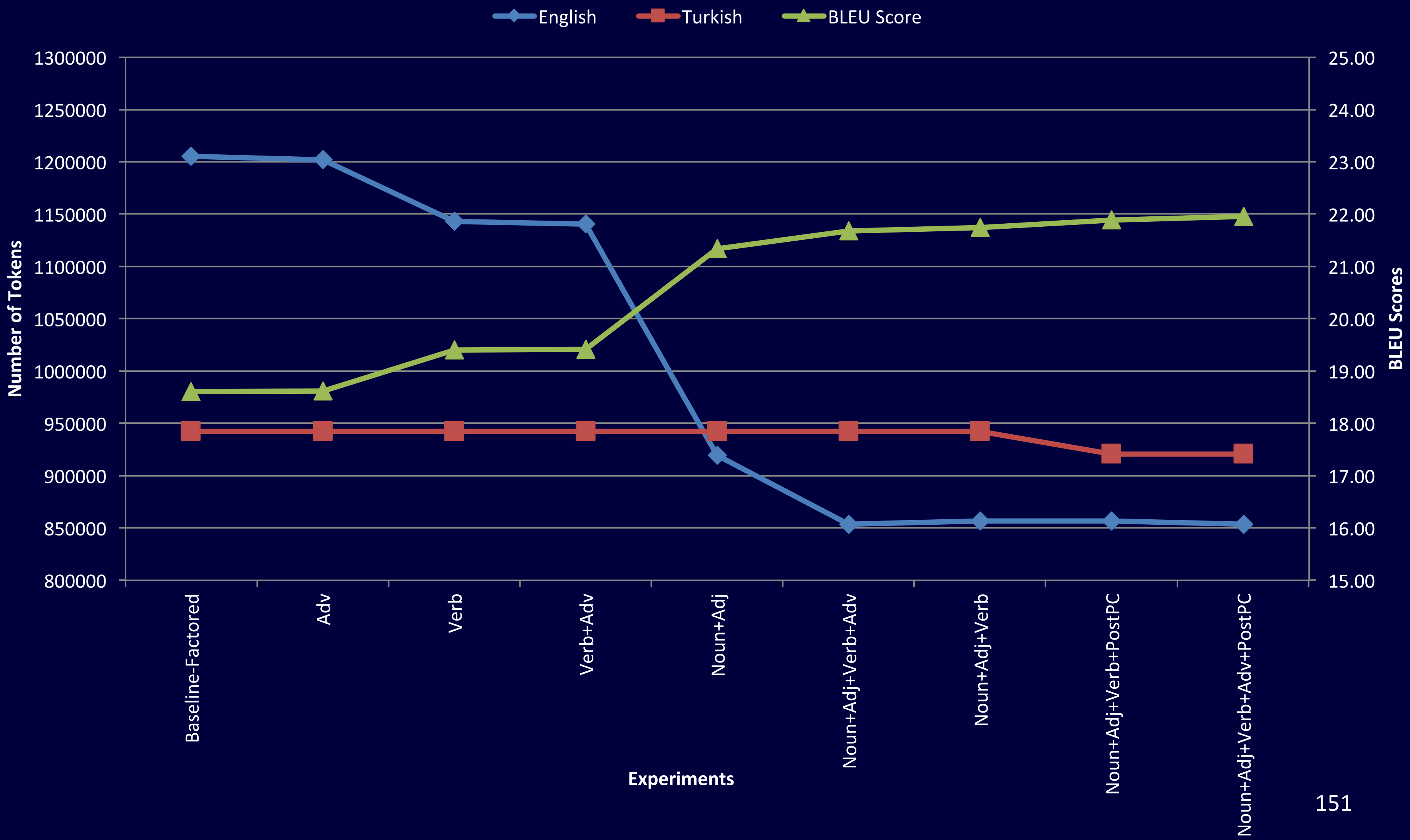
İstek$_3$+Noun sözlü$_6$+Adj ol$_6$+Verb+ByDoingSo yap$_5$+Verb+Pass+Narr+Cond

yetkili$_8$+Adj makam$_8$+Noun bu$_{14}$+Pron+Acc kaydet$_{10,12}$+Verb+Neces+Cop

Syntax and Morphology Interaction

- Transformations on the English side reduce sentence length
- This helps alignment
  - Morphemes and most function words never get involved in alignment
- We can use factored phrase-based translation
  - Phrased-based framework with morphology support

Syntax and Morphology Interaction



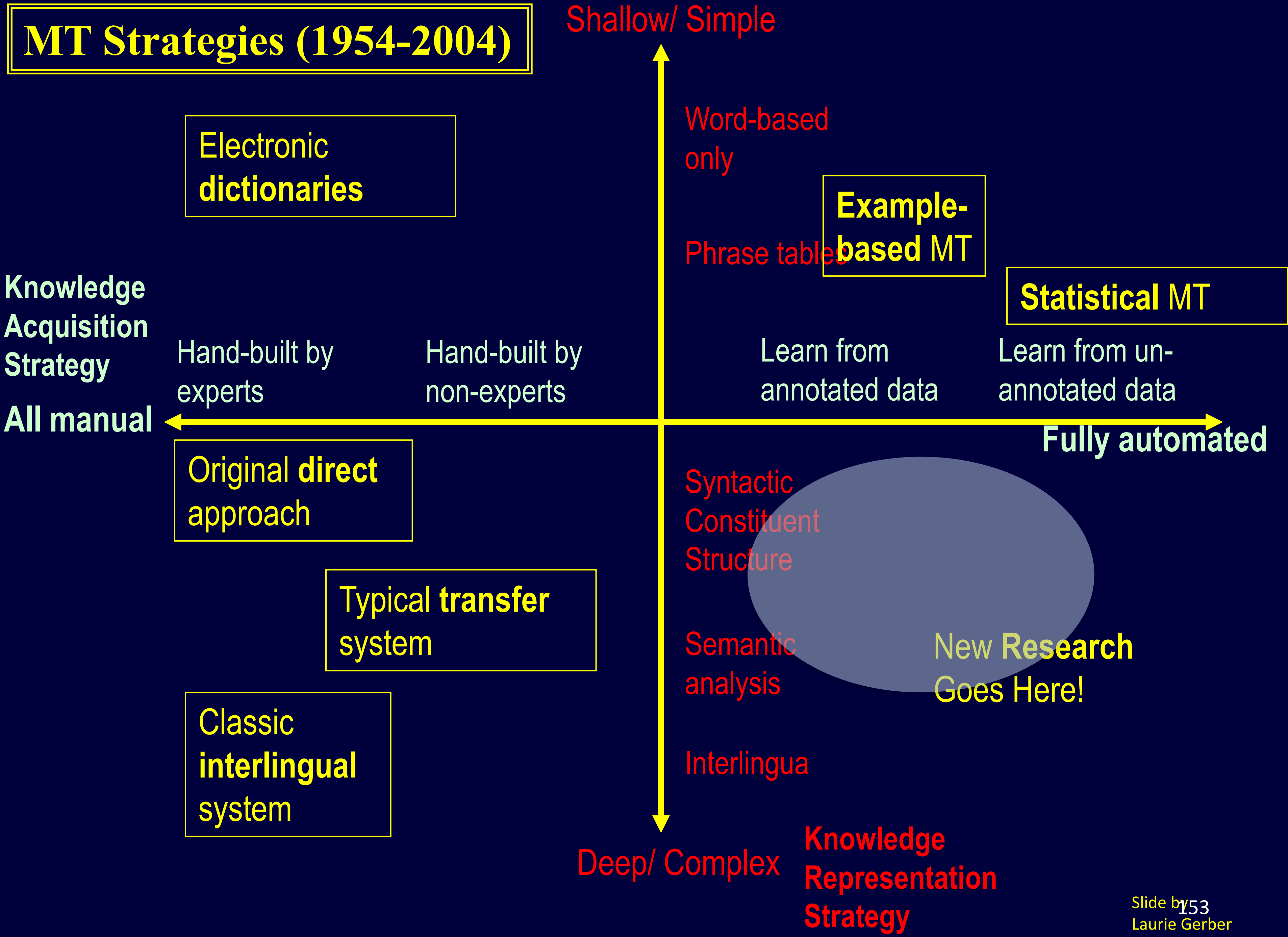
Syntax and Morphology Interaction

- She is reading.
  - She is the subject of read
  - is is the auxiliary of read

- She is read+ing => read +ing+is+she

  taQrAA QrAA +*ta

MT Strategies (1954-2004)



Shallow/ Simple

Word-based only

Phrase tables

Syntactic Constituent Structure

Semantic analysis

Interlingua

Deep/ Complex

Knowledge Representation Strategy

Knowledge Acquisition Strategy

All manual

Hand-built by experts

Hand-built by non-experts

Learn from annotated data

Learn from un-annotated data

Fully automated

Electronic **dictionaries**

Example-based MT

Statistical MT

Original **direct** approach

Typical **transfer** system

Classic **interlingual** system

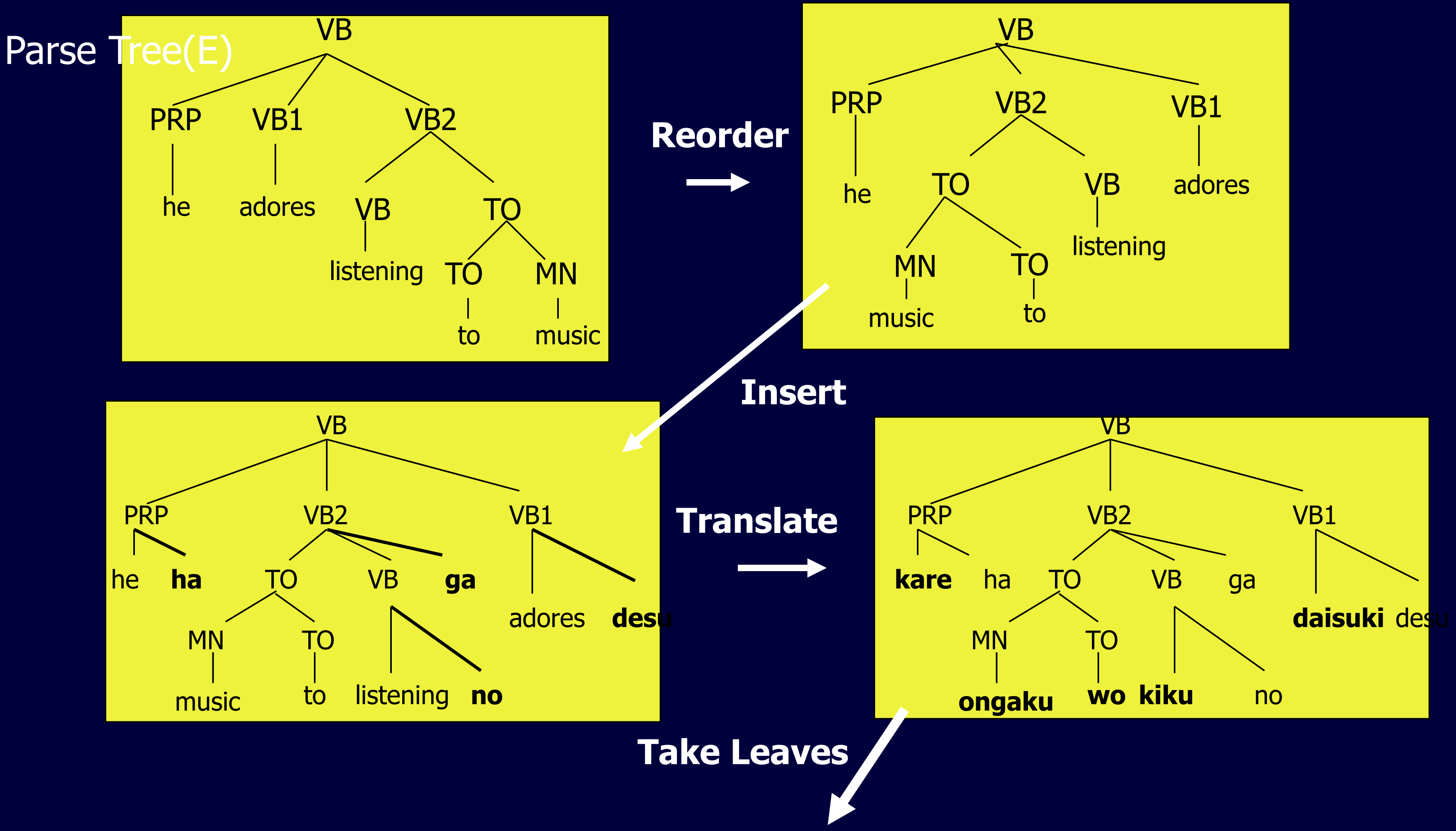
New **Research** Goes Here!

Slide by Laurie Gerber

Syntax in SMT

- Early approaches relied on high-performance parsers for one or both languages
  - Good applicability when English is the source language
    - Tree-to-tree or tree-to-string transductions
- Recent approaches induce synchronous grammars during training
  - Grammar that describe two languages at the same time
    - NP => ADJ_{e1} NP_{e2} : NP_{f2} AD_{Jf1}

Tree-to-String Transformation



Parse Tree(E)

Reorder

Insert

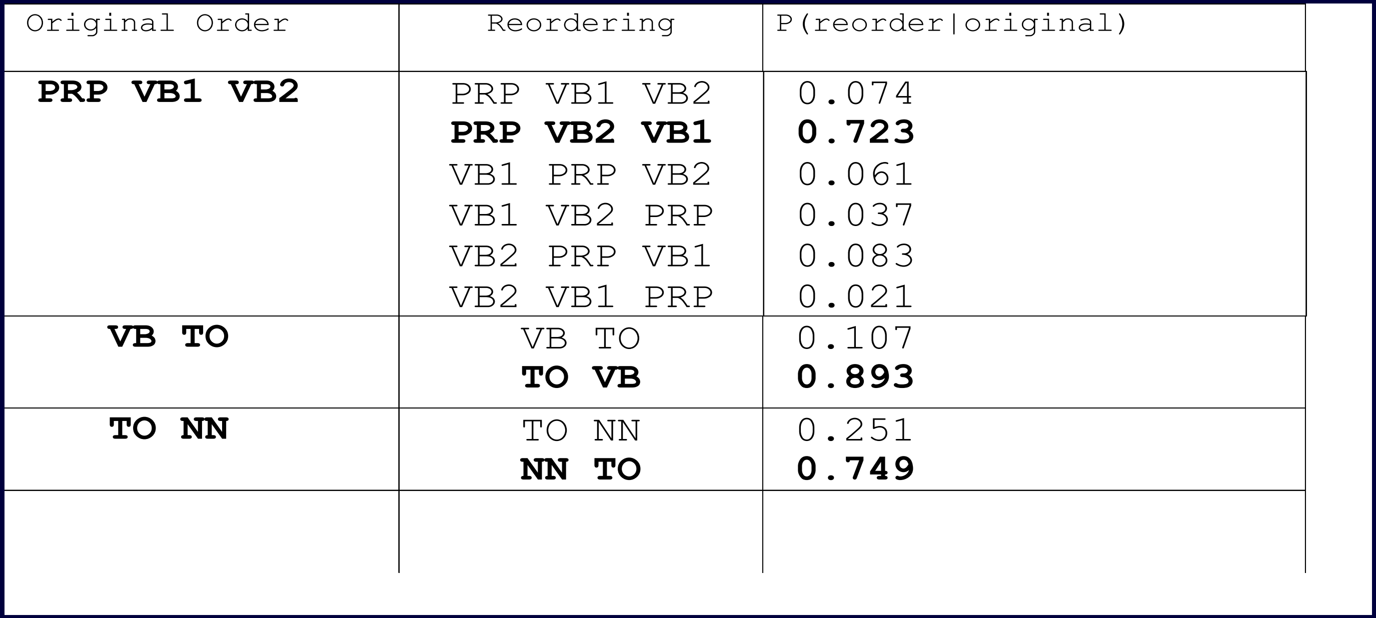
Translate

Take Leaves

Sentence(J) *Kare ha ongaku wo kiku no ga daisuki desu*

Tree-to-String Transformation

- Each step is described by a statistical model
  - Reorder children on a node probabilistically
  - R-table
  - English – Japanese table



Original Order	Reordering	P(reorder|original)
PRP VB1 VB2	PRP VB1 VB2	0.074
	PRP VB2 VB1	**0.723**
	VB1 PRP VB2	0.061
	VB1 VB2 PRP	0.037
	VB2 PRP VB1	0.083
	VB2 VB1 PRP	0.021
VB TO	VB TO	0.107
	TO VB	**0.893**
TO NN	TO NN	0.251
	NN TO	**0.749**

Tree-to-String Transformation

- Each step is described by a statistical model
  - Insert new sibling to the left or right of a node probabilitically
  - Translate source nodes probabilistically

Hierarchical phrase models

- Combines phrase-based models and tree strutures
- Extract synchronous grammars from parallel text
- Uses a statistical chart-parsing algorithm during decoding
  - Parse and generate concurrently

For more info

- Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation (SSST-3) at NAACL HLT 2009
  - http://aclweb.org/anthology-new/W/W09/#2300
- Proceedings of the ACL-08: HLT Second Workshop on Syntax and Structure in Statistical Translation (SSST-2)
  - http://aclweb.org/anthology-new/W/W08/#0400

Acknowledgments

- Some of the tutorial material is based on slides by
  - Kevin Knight (USC/ISI)
  - Philipp Koehn (Edinburgh)
  - Reyyan Yeniterzi (CMU/LTI)

Important References

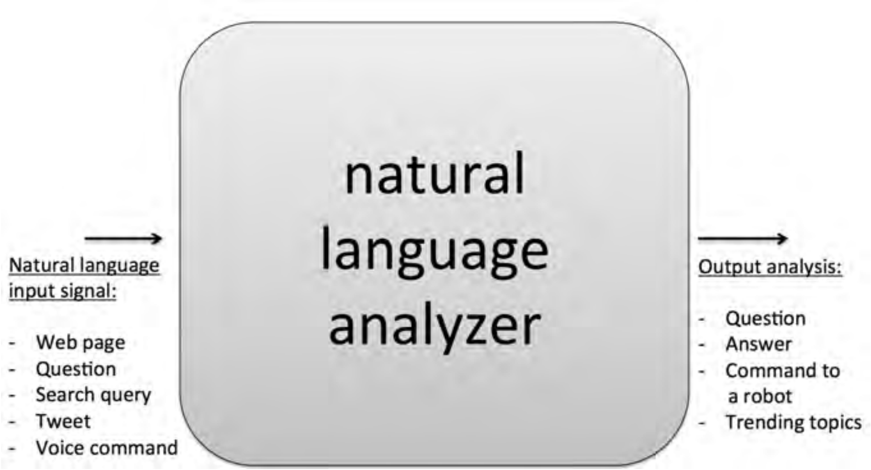
- Statistical Machine Translation (2010)
  - Philipp Koehn
  - Cambridge University Press
- SMT Workbook (1999)
  - Kevin Knight
  - Unpublished manuscript at http://www.isi.edu/~knight/
- http://www.statmt.org
- http://aclweb.org/anthology-new/
  - Look for “Workshop on Statistical Machine Translation”

11-411
Natural Language Processing
Neural Networks and Deep Learning in NLP

Kemal Oflazer

Carnegie Mellon University in Qatar

Big Picture: Natural Language Analyzers



natural language analyzer

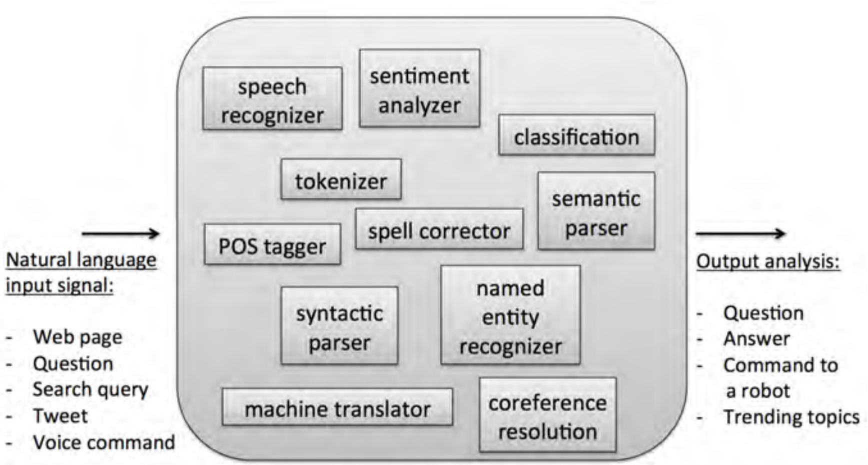
Natural language input signal:

- Web page
- Question
- Search query
- Tweet
- Voice command

Output analysis:

- Question
- Answer
- Command to a robot
- Trending topics

Big Picture: Natural Language Analyzers



speech recognizer

sentiment analyzer

classification

tokenizer

semantic parser

POS tagger

spell corrector

syntactic parser

named entity recognizer

machine translator

coreference resolution

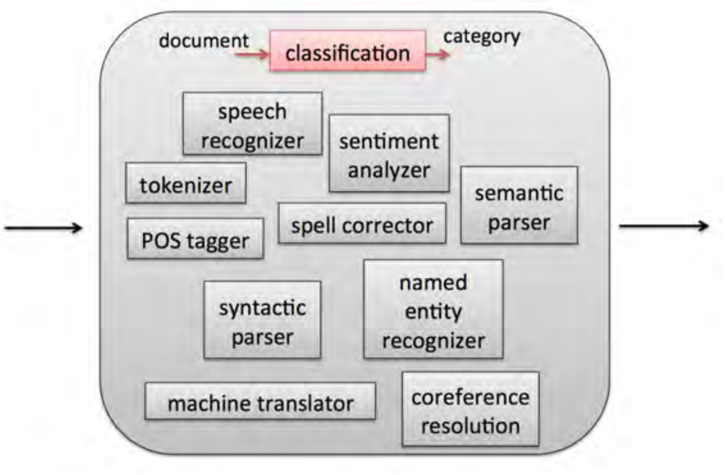
Natural language input signal:

- Web page
- Question
- Search query
- Tweet
- Voice command

Output analysis:

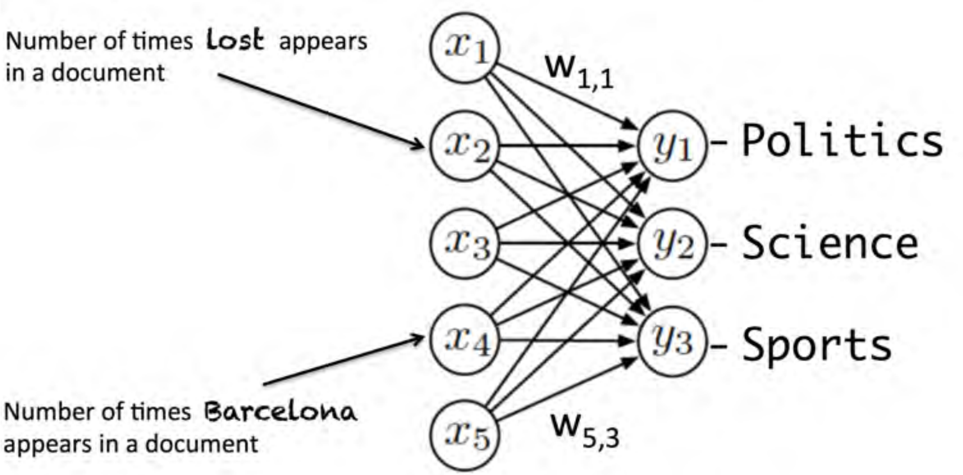
- Question
- Answer
- Command to a robot
- Trending topics

Big Picture: Natural Language Analyzers



Linear Models

- $y_1 = w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + w_{41}x_4 + w_{51}x_5$


Number of times Lost appears in a document
Number of times Barcelona appears in a document
x_1
x_2
x_3
x_4
x_5
$w_{1,1}$
$w_{5,3}$
y_1 - Politics
y_2 - Science
y_3 - Sports

Perceptrons

- Remember Perceptrons?
- A very simple algorithm guaranteed to eventually find a linear separator hyperplane (determine $\boldsymbol{w}$), if one exists.
- If one doesn't, the perceptron will oscillate!
- Assume our classifier is

$$\texttt{classify}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}) > 0 \\ 0 & \text{if } \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}) \leq 0 \end{cases}$$

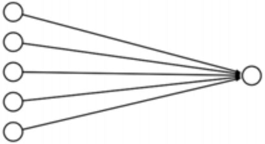
- Start with $\boldsymbol{w} = \mathbf{0}$
- for $t = 1, \ldots, T$
  - $i = t \mod N$
  - $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \left(\ell_i - \texttt{classify}(\boldsymbol{x}_i)\right) \boldsymbol{\Phi}(\boldsymbol{x}_i)$
- Return $\boldsymbol{w}$
- α is the *learning rate* – determined by experimentation.

Perceptrons

- For classification we are basically computing

$$score(\boldsymbol{x}) = \boldsymbol{W} \times \boldsymbol{f}(\boldsymbol{x})^T = \sum_j w_j \cdot f_j(\boldsymbol{x})$$

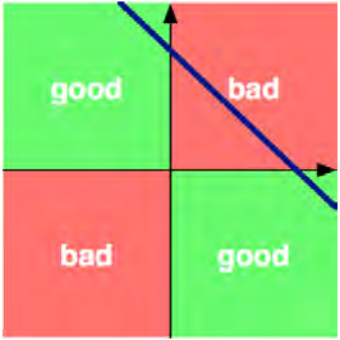
  - w_j are the weights comprising $\boldsymbol{W}$
  - $f_j(\boldsymbol{x})$ are the feature functions.
- We are then deciding based on the value of $score(\boldsymbol{x})$
- Such a computation can be viewed as a “network”.



- Feature function values are provided by the nodes on the left.
- Edges have the weights w_i. Each feature value is multipleid with the respective edge weight.
- The node on the right sums up the incoming values and decides.

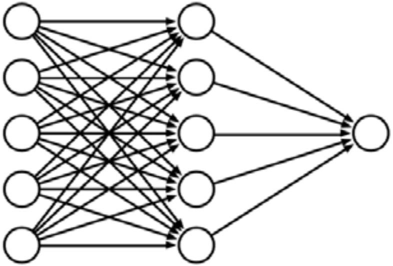
Perceptron

- While quite useful, such a model can only classify “linearly separable” classes.
- So it fails for a very simple problem such as the exclusive-or



Multiple Layers

- We can add an intermediate “hidden” layer.
  - each arrow is a weight



- Have we gained anything?
  - Not really. We have a linear combination of weights (input to hidden and hidden to output),
  - Those two can be combined offline to a single weight matrix.

Adding Non-linearity

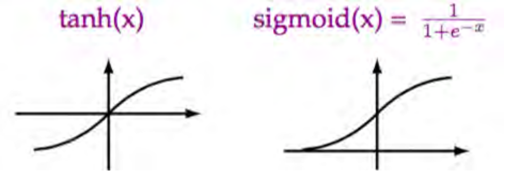
- Instead of computing a linear combination

$$score(\boldsymbol{x}) = \sum_j w_j \cdot f_j(\boldsymbol{x})$$

- We use a non-linear function F

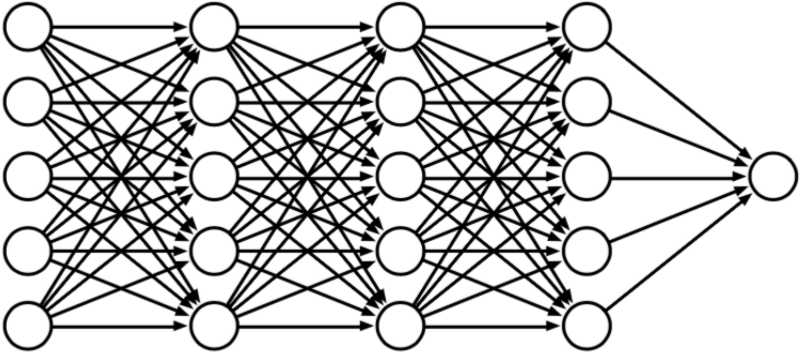
$$score(\boldsymbol{x}) = F\Big(\sum_j w_j \cdot f_j(\boldsymbol{x})\Big)$$

- Some popular choices for F



Deep Learning

- More layers ⇒ “deep learning”

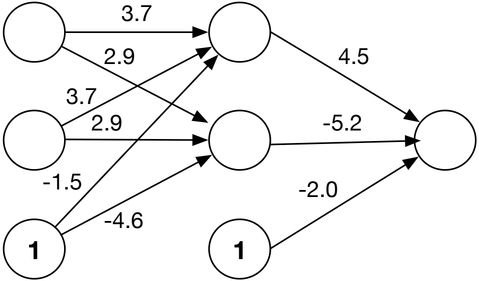


- The sigmoid is also called the “logistic function.”

What Depth Holds

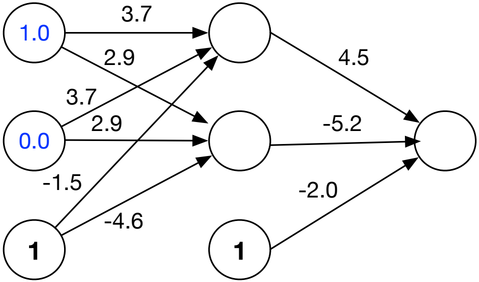
- Each layer is a processing step
- Having multiple processing steps allows complex functions
- Metaphor: NN and computing circuits
  - computer = sequence of Boolean gates
  - neural computer = sequence of layers
- Deep neural networks can implement more complex functions

Simple Neural Network



- One innovation: **bias units** (no input, always value 1)

Sample Input

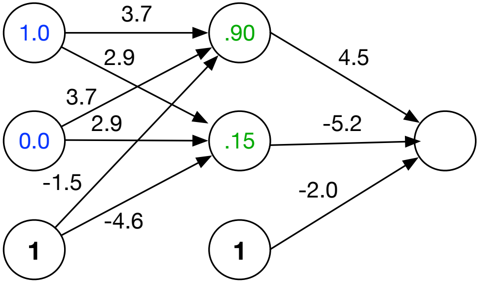


- Try out two input values
- Hidden unit computation

$$sigmoid(1.0 \times 3.7 + 0.0 \times 3.7 + 1 \times -1.5) = sigmoid(2.2) = \frac{1}{1 + e^{-2.2}} = 0.90$$

$$sigmoid(1.0 \times 2.9 + 0.0 \times 2.9 + 1 \times -4.6) = sigmoid(-1.7) = \frac{1}{1 + e^{1.7}} = 0.15$$

Computed Hidden Layer Values

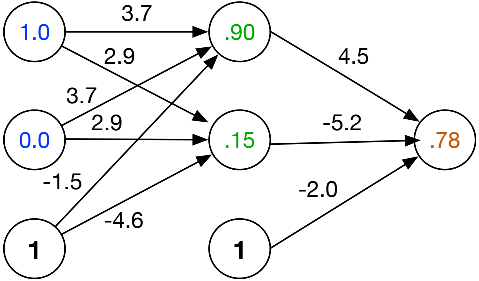


- Try out two input values
- Hidden unit computation

$$sigmoid(1.0 \times 3.7 + 0.0 \times 1.7 + 1 \times -1.5) = sigmoid(2.2) = \frac{1}{1+e^{-2.2}} = 0.90$$

$$sigmoid(1.0 \times 2.9 + 0.0 \times 2.9 + 1 \times -4.5) = sigmoid(-1.7) = \frac{1}{1+e^{1.7}} = 0.15$$

Computed Output Value



- Output unit computation

$$sigmoid(0.90 \times 4.5 + 0.15 \times -5.2 + 1 \times -2.0) = sigmoid(1.25) = \frac{1}{1+e^{-1.25}} = 0.78$$

Output for All Binary Inputs

Input x_0	Input x_1	Hidden h_0	Hidden h_1	Output y_0
0	0	0.18	0.01	$0.23 \rightarrow 0$
0	1	0.90	0.15	$0.78 \rightarrow 1$
1	0	0.90	0.15	$0.78 \rightarrow 1$
1	1	0.99	0.77	$0.18 \rightarrow 0$

- Network implements the XOR
  - hidden node h_0 is OR
  - hidden node h_1 is AND
  - final layer is (essentially) $h_0 - (h_1)$

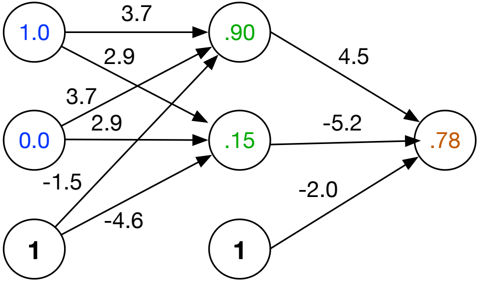
The Brain vs. Artificial Neural Networks

- Similarities
    - Neurons, connections between neurons
    - Learning = change of connections, not change of neurons
    - Massive parallel processing
- But artificial neural networks are much simpler
    - computation within neuron vastly simplified
    - discrete time steps
    - typically some form of supervised learning with massive number of stimuli

Backpropagation Training

- Lather – take an input and run it forward through the network
- Rinse – compare it to the expected output, and adjust weights if they differ
- Repeat – for the next input, until convergence or time-out

Backpropagation Training

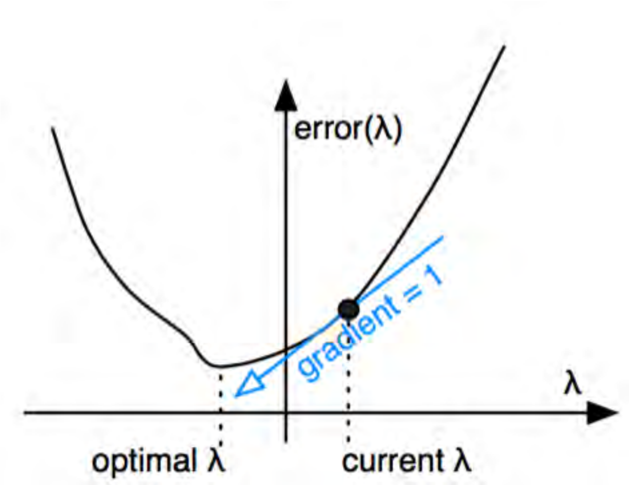


- Computed output is $y = 0.78$
- Correct output is $t(arget) = 1.0$
- How do we adjust the weights?

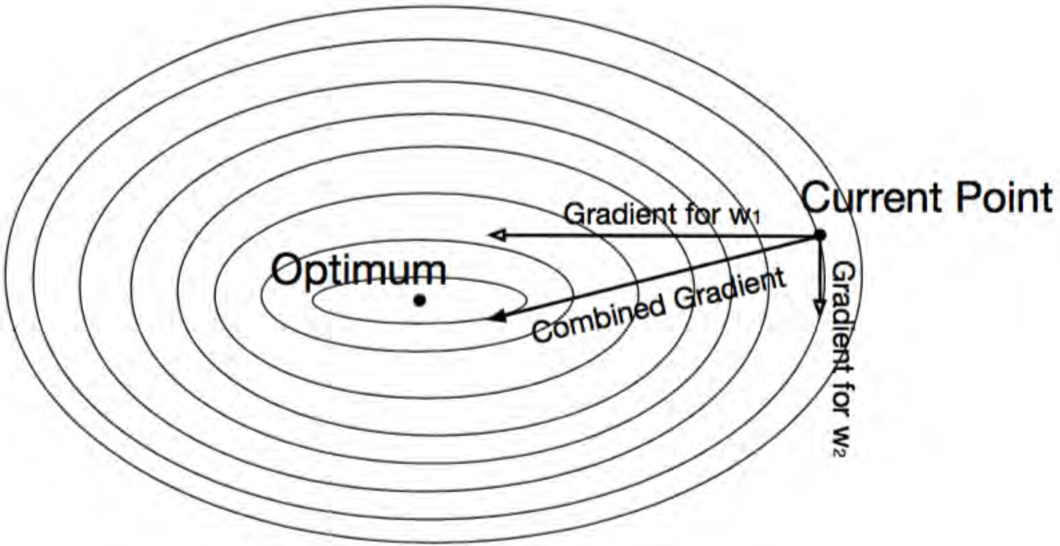
Key Concepts

- Gradient Descent
  - error is a function of the weights
  - we want to reduce the error
  - gradient descent: move towards the error minimum
  - compute gradient → get direction to the error minimum
  - adjust weights towards direction of lower error
- Backpropagation
  - first adjust last set of weights
  - propagate error back to each previous layer
  - adjust their weights

Gradient Descent



Gradient Descent



Derivative of the Sigmoid

- Sigmoid: $$sigmoid(x) = \frac{1}{1+e^{-x}}$$
- Reminder: quotient rule $$(\frac{f(x)}{g(x)})' = \frac{g(x)f'(x) - f(x)g'(x)}{g(x)^2}$$
- Derivative $$\begin{aligned} \frac{d\ sigmoid(x)}{dx} &= \frac{d}{dx}\frac{1}{1+e^{-x}} \\ &= \frac{0 \times (1-e^{-x}) - (-e^{-x})}{(1+e^{-x})^2} \\ &= \frac{1}{1+e^{-x}}(\frac{e^{-x}}{1+e^{-x}}) \\ &= \frac{1}{1+e^{-x}}(1 - \frac{1}{1+e^{-x}}) \\ &= sigmoid(x)(1 - sigmoid(x)) \end{aligned}$$

Final Layer Update (1)

- We have a linear combination of weights and hidden layer values: $s = \sum_k w_k h_k$
- Then we have the activation function: $y = sigmoid(s)$
- We have the error function $E = \frac{1}{2}(t - y)^2$.
  - t is the target ouput.
- Derivative of error with regard to one weight w_k (using chain rule)

$$\frac{dE}{dw_k} = \frac{dE}{dy}\frac{dy}{ds}\frac{ds}{dw_k}$$

- Error is already defined in terms of y, hence

$$\frac{dE}{dy} = \frac{d}{dy}\frac{1}{2}(t - y)^2 = -(t - y)$$

Final Layer Update (2)

- We have a linear combination of weights and hidden layer values: $s = \sum_k w_k h_k$
- Then we have the activation function: $y = sigmoid(s)$
- We have the error function $E = \frac{1}{2}(t - y)^2$.
- Derivative of error with regards to one weight w_k (using chain rule)

$$\frac{dE}{dw_k} = \frac{dE}{dy} \frac{dy}{ds} \frac{ds}{dw_k}$$

- y with respect to s is $sigmoid(s)$

$$\frac{dy}{ds} = \frac{d \; sigmoid(s)}{ds} = sigmoid(s)(1 - sigmoid(s)) = y(1 - y)$$

Final Layer Update (3)

- We have a linear combination of weights and hidden layer values: $s = \sum_k w_k h_k$
- Then we have the activation function: $y = sigmoid(s)$
- We have the error function $E = \frac{1}{2}(t-y)^2$.
- Derivative of error with regards to one weight w_k (using chain rule)

$$\frac{dE}{dw_k} = \frac{dE}{dy}\frac{dy}{ds}\frac{ds}{dw_k}$$

- s is a weighted linear combination of hidden node values h_k

$$\frac{ds}{dw_k} = \frac{d}{dw_k}(\sum_k w_k h_k) = h_k$$

Putting it All Together

- Derivative of error with regard to one weight w_k

$$\frac{dE}{dw_k} = \frac{dE}{dy}\frac{dy}{ds}\frac{ds}{dw_k} = -(t-y)\; y(1-y)\; h_k$$

  - error
  - derivative of sigmoid: y'

- We adjust the weight as follows

$$\Delta w_k = \mu\,(t-y)\,y'\,h_k$$

  where μ is a fixed learning rate.

Multiple Output Nodes

- Our example had one ouput node.
- Typically neural networks have multiple output nodes.
- Error is computed over all j output nodes

$$E = \frac{1}{2} \sum_j (t_j - y_j)^2$$

- Weight w_{kj} from hidden unit k to output unit j is adjusted according to node j

$$\Delta w_{kj} = \mu \, (t_j - y_j) \, y_j' \, h_k$$

- We can also rewrite this as

$$\Delta w_{kj} = \mu \, \delta_j \, h_k$$

where δ_j is the **error term** for output unit j.

Hidden Layer Update

- In a hidden layer, we do not have a target output value.
- But we can compute how much each hidden node contributes to the downstream error E.
  - k refers to a hidden node
  - j refers to a node in the next/output layer
- Remember the error term

$$\delta_j = (t_j - y_j)y'_j$$

- The error term associated with hidden node k is (skipping the multivariate math) is

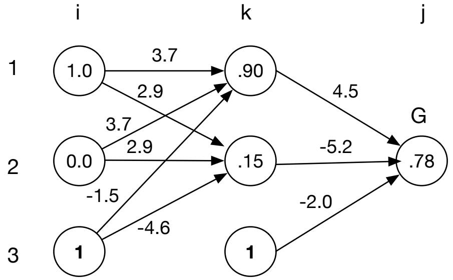
$$\delta_k = (\sum_j w_{kj}\delta_j)h'_k$$

- So if the u_{ik} is the weight between input unit x_i and hidden unit k then

$$\Delta u_{ik} = \mu\, \delta_k\, x_i$$

- Compare with $\Delta w_{kj} = \mu\, \delta_j\, h_k$.

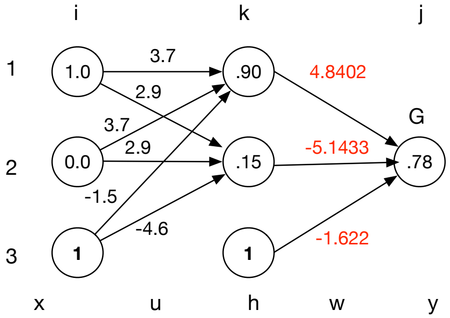
An Example



For the output unit G

- Computed output = $y_1 = 0.78$
- Correct output = $t = 1.0$
- Final layer weight updates (with learning rate $\mu = 10$)
  - $\delta_1 = (t_1 - y_1)y_1' = (1 - 0.78) \times 0.172 = 0.0378$
  - $\Delta w_{11} = \mu\delta_1 h_1 = 10 \times 0.0378 \times 0.90 = 0.3402$
  - $\Delta w_{21} = \mu\delta_1 h_2 = 10 \times 0.0378 \times 0.15 = 0.0567$
  - $\Delta w_{31} = \mu\delta_1 h_3 = 10 \times 0.0378 \times 1 = 0.378$

An Example



4.5+0.3402=4.8402

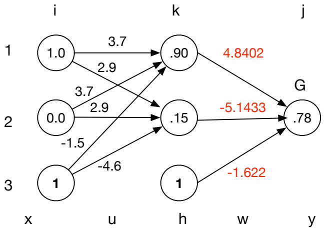
-5.2+0.0.0567=-5.1433

-2.0+0.378=-1.622

For the output unit G

- Computed output = $y_1 = 0.78$
- Correct output = $t = 1.0$
- Final layer weight updates (with learning rate $\mu = 10$)
  - $\delta_1 = (t_1 - y_1)y_1' = (1 - 0.78) \times 0.172 = 0.0378$
  - $\Delta w_{11} = \mu \delta_1 h_1 = 10 \times 0.0378 \times 0.90 = 0.3402$
  - $\Delta w_{21} = \mu \delta_1 h_2 = 10 \times 0.0378 \times 0.15 = 0.0567$
  - $\Delta w_{31} = \mu \delta_1 h_3 = 10 \times 0.0378 \times 1 = 0.378$

Hidden Layer Updates



4.5+0.3402=4.8402

-5.2+0.0.0567=-5.1433

-2.0+0.378=-1.622

For hidden unit h_1

- $\delta_1 = (\sum_j w_{1j}\,\delta_1^G)h_1' = 4.5 \times 0.0378 \times 0.09 = 0.015$
- $\Delta u_{11} = \mu\delta_1 x_1 = 10 \times 0.015 \times 1.0 = 0.175$
- $\Delta u_{21} = \mu\delta_1 x_2 = 10 \times 0.015 \times 0.0 = 0$
- $\Delta u_{31} = \mu\delta_1 x_3 = 10 \times 0.015 \times 1.0 = 0.175$

Repeat for hidden unit h_2

- $\delta_2 = (\sum_j w_{2j}\,\delta_1^G)h_2' = -5.2 \times 0.0378 \times 0.1275 = -0.025$
- $\Delta u_{12} = \ldots$
- $\Delta u_{22} = \ldots$
- $\Delta u_{32} = \ldots$

Initialization of Weights

- Random initialization e.g., uniformly in the interval

$$[-0.01, 0.01]$$

- For shallow networks there are suggestions for

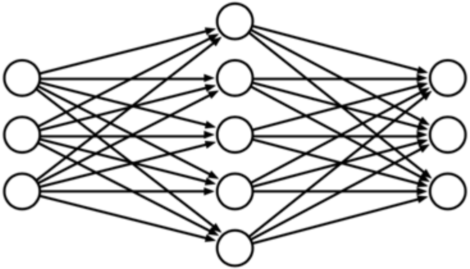
$$[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}]$$

- For deep networks there are suggestions for

$$[-\frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}]$$

where n_i and n_{i+1} are sizes of the previous and next layers.

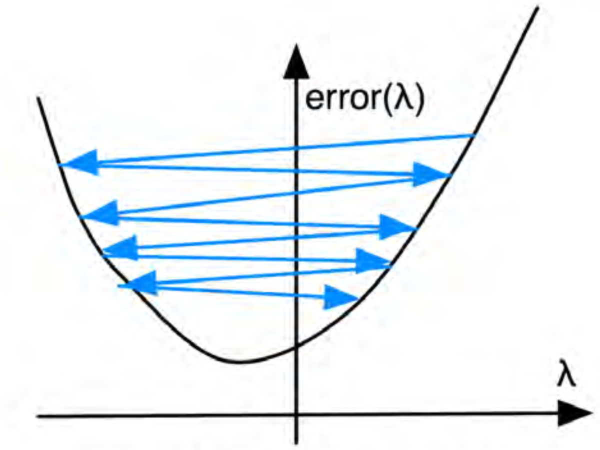
Neural Networks for Classification



- Predict Class: one output per class
- Training data output is a "one-hot-vector", e.g., $\boldsymbol{y} = [0, 0, 1]^T$
- Prediction:
  - predicted class is output node i with the highest value y_i
  - obtain posterior probability distribution by softmax

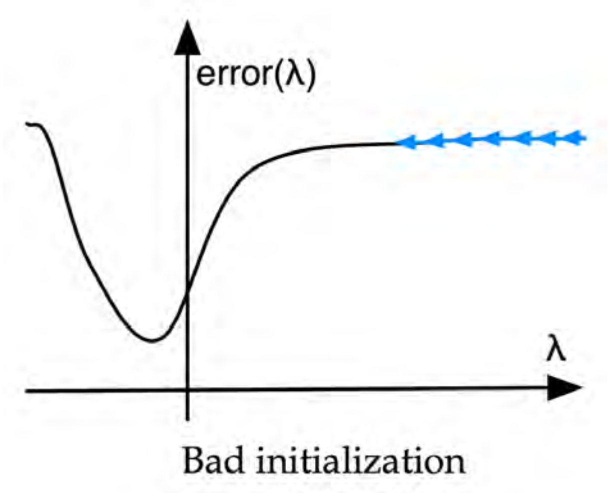
$$softmax(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Problems with Gradient Descent Training

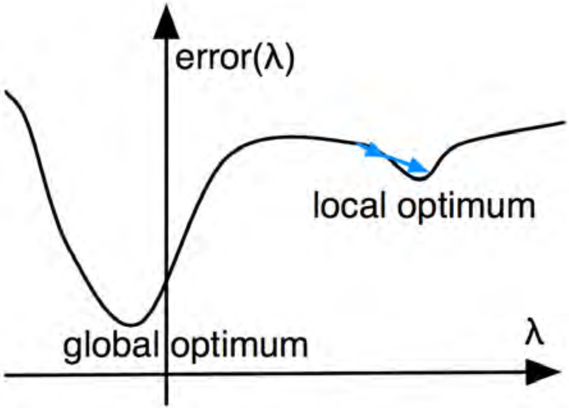


Too high learning rate

Problems with Gradient Descent Training



Problems with Gradient Descent Training



Local optimum

Speed-up: Momentum

- Updates may move a weight slowly in one direction
- We can keep up a memory if prior updates

$$\Delta w_{kj}(n-1)$$

- and add these to any new updates with a decay factor ρ

$$\Delta w_{kj}(n) = \mu\,\delta_j\,h_k + \rho\Delta w_{kj}(n-1)$$

Dropout

- A general problem of machine learning: overfitting to training data (very good on train, bad on unseen test)
- Solution: **regularization**, e.g., keeping weights from having extreme values
- Dropout: randomly remove some hidden units during training
  - mask: set of hidden units dropped
  - randomly generate, say, 10 – 20 masks
  - alternate between the masks during training

Mini Batches

- Each training example yields a set of weight updates Δw_{ij}
- Batch up several training examples
  - Accumulate their updates
  - Apply sum to the model – one big step instead of many small steps
- Mostly done for speed reasons

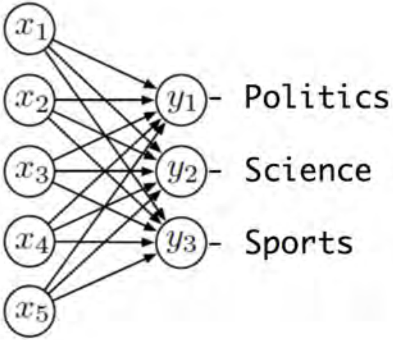
Matrix Vector Formulation

- Forward computation $\boldsymbol{s} = \boldsymbol{W}\,\boldsymbol{h}$
- Activation computation $\boldsymbol{y} = sigmoid(\boldsymbol{s})$
- Error Term: $\boldsymbol{\delta} = (\boldsymbol{t} - \boldsymbol{y}) \cdot sigmoid'(\boldsymbol{s})$
- Propagation of error term: $\boldsymbol{\delta}_i = \boldsymbol{W}\boldsymbol{\delta}_{i+1} \cdot sigmoid'(\boldsymbol{s})$
- Weight updates: $\Delta\boldsymbol{W} = \mu\,\boldsymbol{\delta}\,\boldsymbol{h}^T$

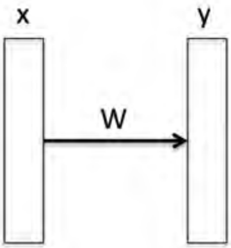
Toolkits

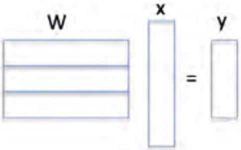
- Theano (Python Library)
- Tensorflow (Python Library, Google)
- PyTorch (Python Library, Facebook)
- MXNet (Python Library, Amazon)
- DyNet (Python Library, A consortium of institutions including CMU)

Neural Network V1.0: Linear Model



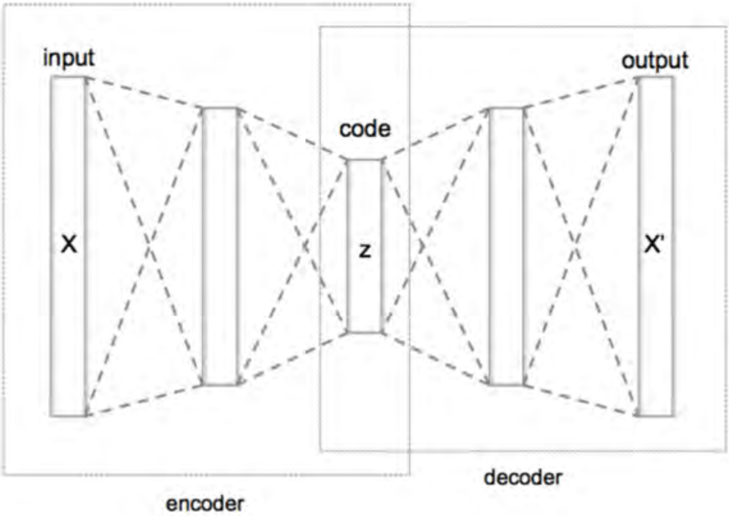
same as





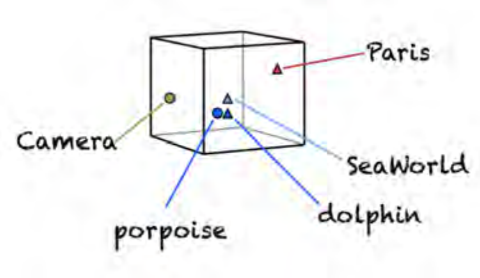
Neural Network v2.0: Representation Learning

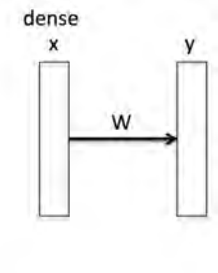
- Big idea: induce low-dimensional dense feature representations of high-dimensional objects



Neural Network v2.1: Representation Learning

- Big idea: induce low-dimensional dense feature representations of high-dimensional objects

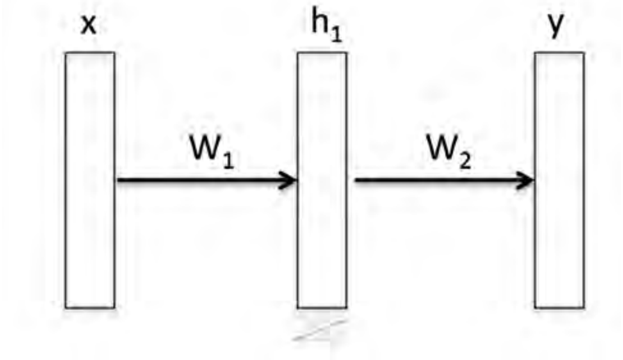




- Did this really solve the problem?

Neural Network v3.0: Complex Functions

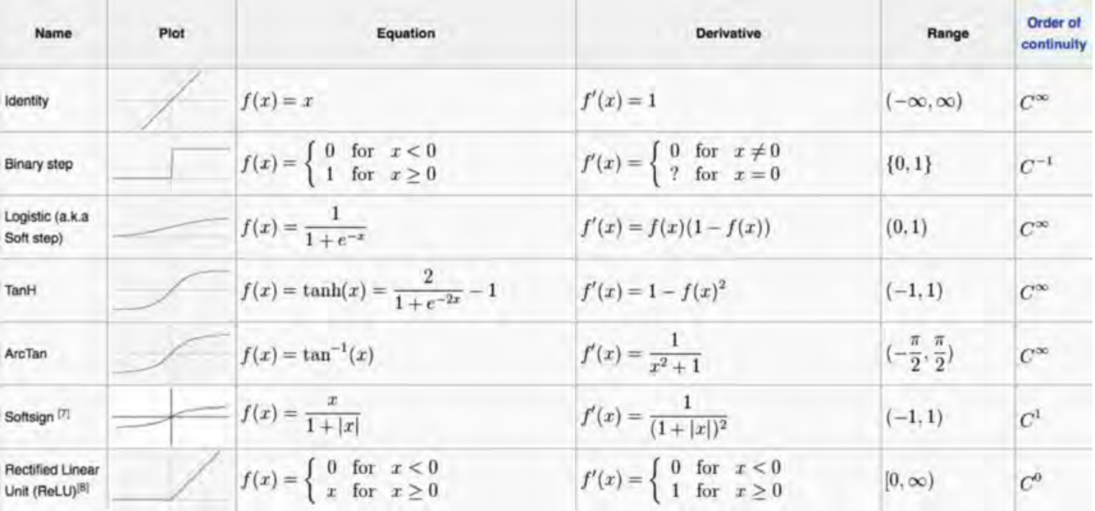
- Big idea: define complex functions by adding a hidden layer.



- $\boldsymbol{y} = \boldsymbol{W}_2\boldsymbol{h}_1 = a_1(\boldsymbol{W}_1\boldsymbol{x}_1)$

Neural Network v3.0: Complex Functions

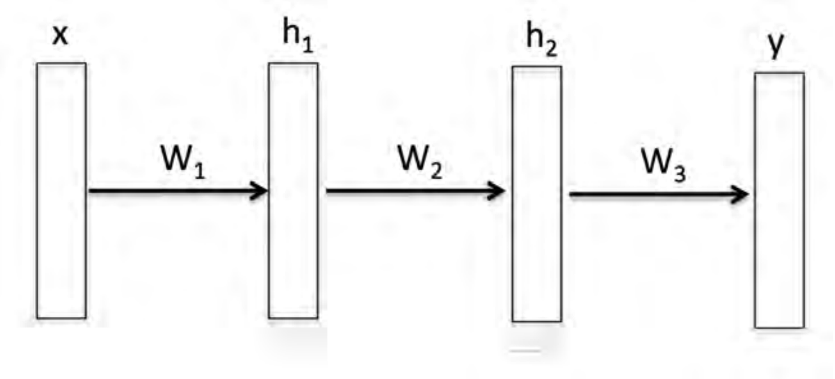
- Popular activation/transfer/non-linear functions



Name	Plot	Equation	Derivative	Range	Order of continuity
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$	C^∞
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$	$\{0, 1\}$	C^{-1}
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$	C^∞
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^∞
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$(-\frac{\pi}{2}, \frac{\pi}{2})$	C^∞
Softsign [7]		$f(x) = \frac{x}{1 + \lvert x \rvert}$	$f'(x) = \frac{1}{(1 + \lvert x \rvert)^2}$	$(-1, 1)$	C^1
Rectified Linear Unit (ReLU)[8]		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$[0, \infty)$	C^0

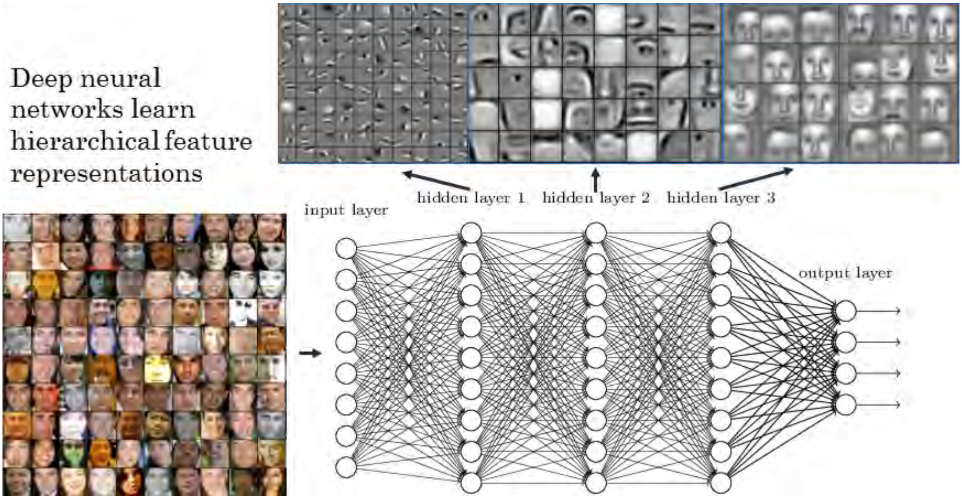
Neural Network v3.5: Deeper Networks

- Add more layers!



- $\boldsymbol{y} = \boldsymbol{W}_3\boldsymbol{h_2} = \boldsymbol{W}_3 a_2(\boldsymbol{W}_2(a_1(\boldsymbol{W}_1\boldsymbol{x}_1))$

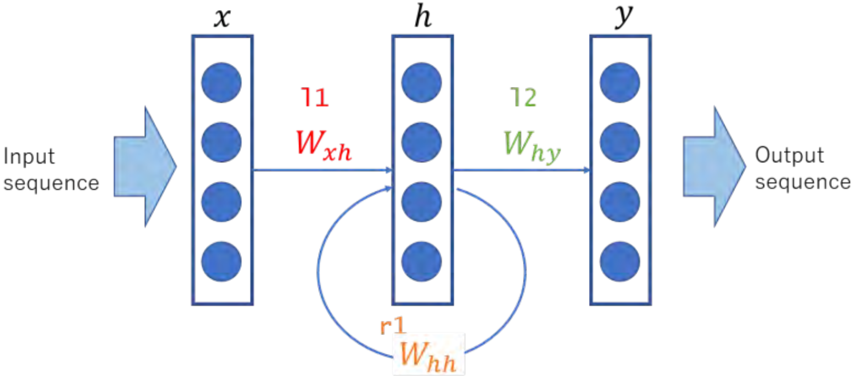
Neural Network v3.5: Deeper Networks



Deep neural networks learn hierarchical feature representations

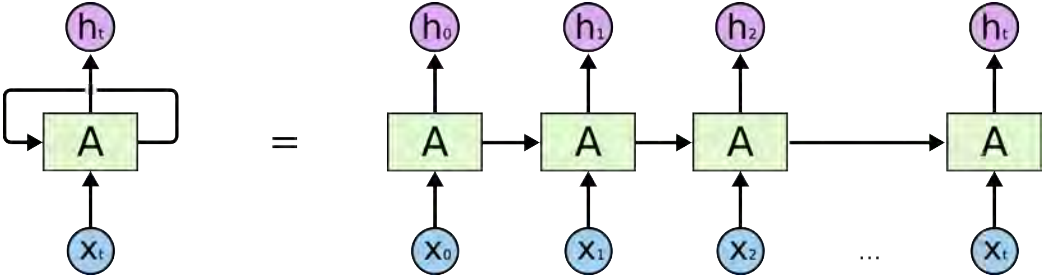
Neural Network v4.0: Recurrent Neural Networks

- Big Idea: Use hidden layers to represent sequential state



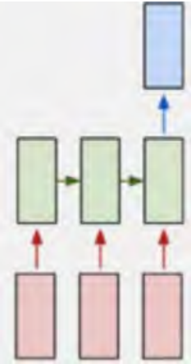
$$h_t = \tanh(W_{xh} x_t + W_{hh} h_{t-1})$$
$$y_t = W_{hy} h_t$$

Neural Network v4.0: Recurrent Neural Networks

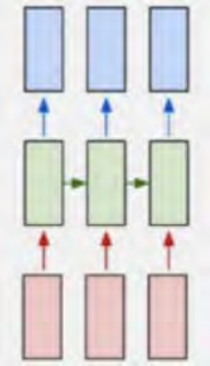


Neural Network v4.1: Output Sequences

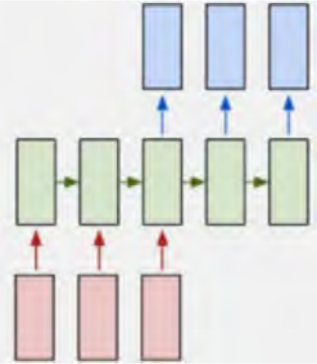
Many to One



Many to Many

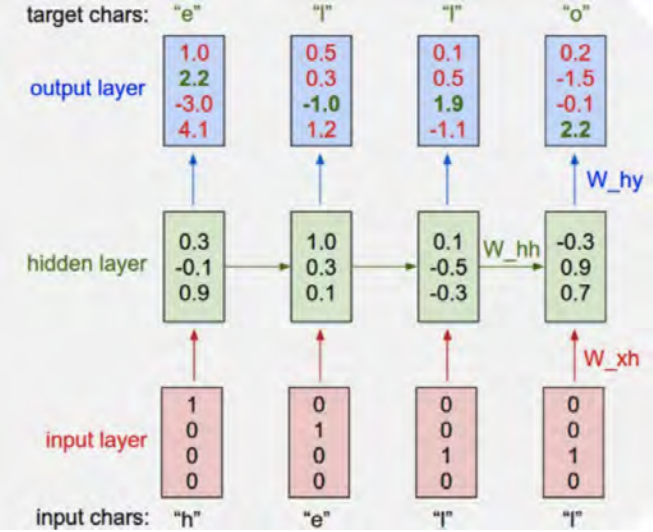


Many to Many



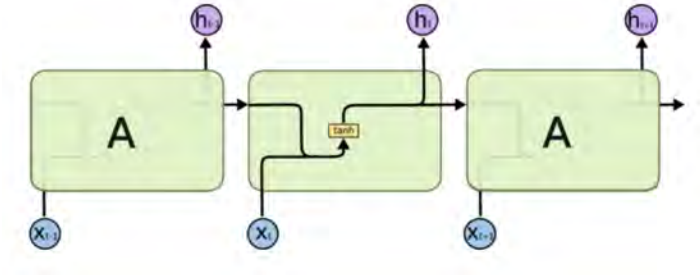
Neural Network v4.1: Output Sequences

- Character-level Language Models

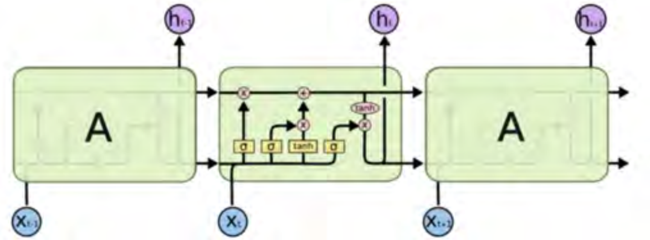


Neural Network v4.2: Long-Short Term Memory

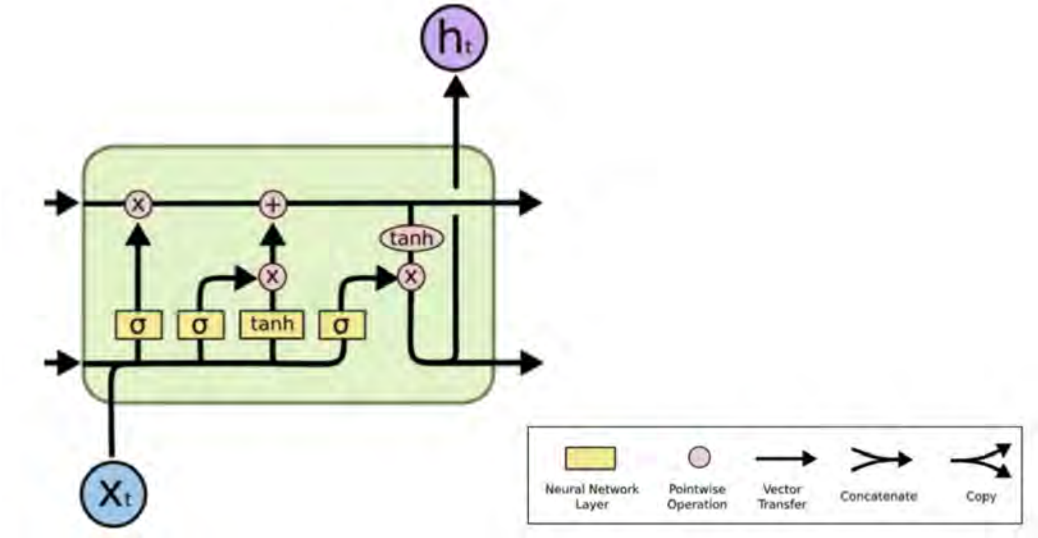
- Regular Recurrent Networks



- LSTMs

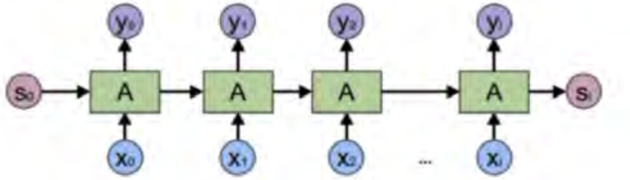


Neural Network v4.2: Long-Short Term Memory

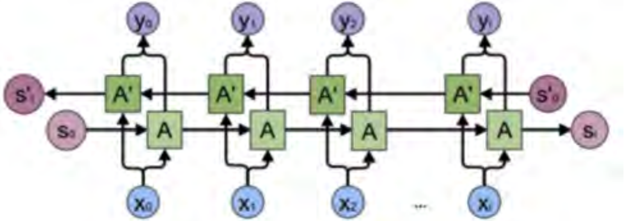


Neural Network v4.3: Bidirectional RNNs

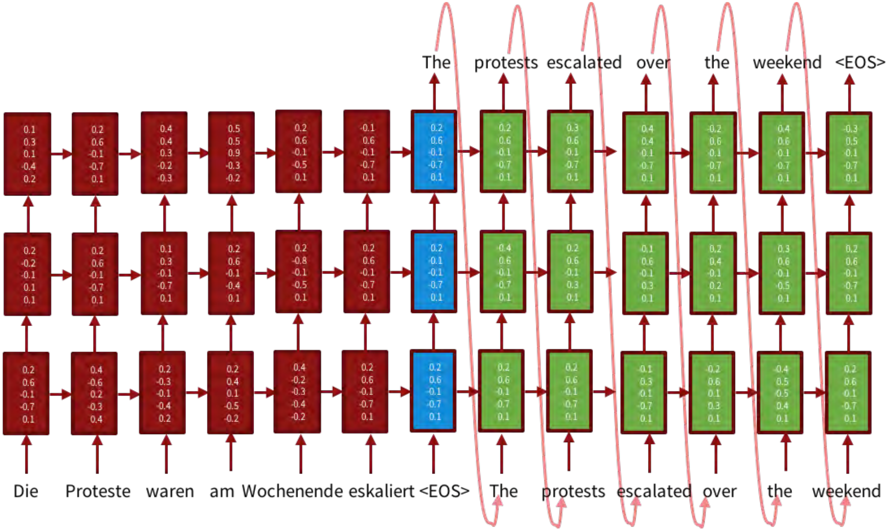
- Unidirectional RNNs



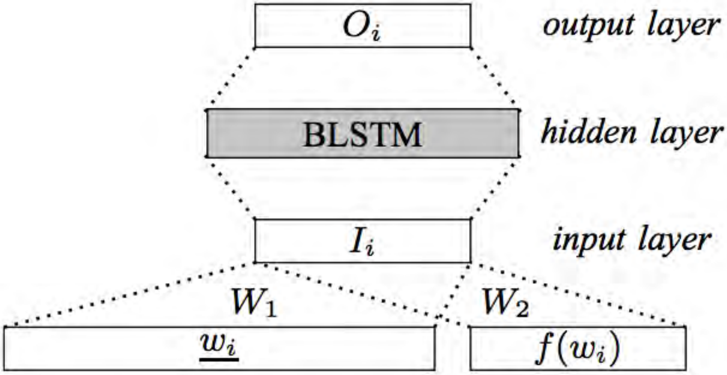
- Bidirectional RNNs



Neural Machine Translation



Neural Part-of-Speech Tagging



- w_i is the one-hot representation of the current word.
- $f(w_i)$ encodes the case of of w_i: all caps, cap initial, lowercase.

Neural Parsing

Softmax layer:

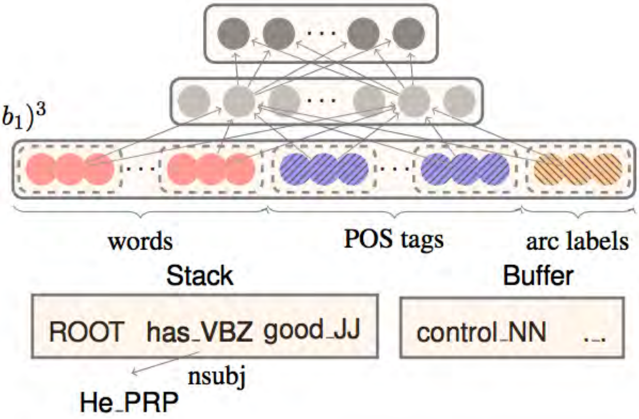
$$p = \texttt{softmax}(W_2 h)$$

Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Input layer: $[x^w, x^t, x^l]$

Configuration



words
POS tags
arc labels
Stack
Buffer
ROOT has_VBZ good_JJ
control_NN ...
nsubj
He_PRP