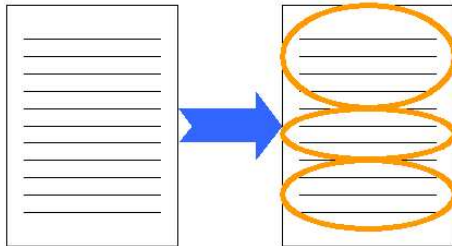


Text Segmentation

Text Segmentation

Goal: Partition a text into a sequence of topically coherent blocks



Applications: Information Retrieval, Summarization, Question Answering

Example

So -- last time we talked about propositional logic. There's no better way to empty out a room than to talk about logic. So now, today we're going to talk about what it is that you might -- having done -- gone to all that work of establishing syntax and semantics and all that -- what might you actually want to do with some descriptions that are written down in logic? So there are two things that we might want to automatically determine about a sentence of logic. Well, and maybe there are others but one is satisfiability, and another is validity. OK. We -- this is a test for you guys -- last time we talked about a way to determine whether a sentence is satisfiable. Can you tell me what it is? You know an algorithm for this. Yes? It could be possible to find the variables that make it true? Right. So it's satisfiable if there's some assignment that makes it true. And so you could obviously -- and you read all the assignments and see if there's one that makes it true. And how do you tell if a sentence is valid? Anybody else? So the same thing but except all of them. So valid means it's true in every assignment. Satisfiable means there's one assignment that makes it true; validity, every assignment makes it true. So, we're going to next talk about better ways to compute satisfiability and better ways to compute validity. That's going to be our theme for today and maybe some more of tomorrow, I'm not sure. So, satisfiability problems -- it turns out that there are cases that -- there are problems in the real world that end up being expressed essentially as lists of constraints where you're trying to find some, say, assignment of values to variables that satisfy the constraints. So an example might be scheduling people to work shifts in a hospital, right? Filling out the nurse shifts in a hospital. Different people have different constraints, some don't want to work at night, no individual can work more than this many hours out of that many hours, these two people don't want to be on the same shift, you have to have at least this many per shift and so on. So you can often describe a setting like that as a bunch of constraints on a set of variables. There's an interesting application of satisfiability that's going on here at MIT in the Lab for Computer Science, in fact I want to put a link to Daniel Jackson's home page, maybe you can help me to remember to do that. So Professor Jackson's doing this thing where he's interested in trying to find bugs in programs. So that's a good thing to do, but he wants to get the computer to do it automatically. And one way to do it is to essentially make a small example instance of a program. So an example of a kind of program that he might want to try to find a bug in would be an air traffic controller. So there's -- the air traffic controller has all these rules about how it works, right?

So -- last time we talked about propositional logic. There's no better way to empty out a room than to talk about logic. So now, today we're going to talk about what it is that you might -- having done -- gone to all that work of establishing syntax and semantics and all that -- what might you actually want to do with some descriptions that are written down in logic? So there are two things that we want to automatically determine about a sentence of logic. Well, and maybe there are others but one is satisfiability, and another is validity. OK. We -- this is a test for you guys -- last time we talked about a way to determine whether a sentence is satisfiable. Can you tell me what it is? You know an algorithm for this. Yes? It could be possible to find the variables that make it true? Right. So it's satisfiable if there's some assignment that makes it true. And so you could obviously -- and you read all the assignments and see if there's one that makes it true. And how do you tell if a sentence is valid? Anybody else? So the same thing but except all of them. So valid means it's true in every assignment. Satisfiable means there's one assignment that makes it true; validity, every assignment makes it true. So, we're going to next talk about better ways to compute satisfiability and better ways to compute validity. That's going to be our theme for today and maybe some more of tomorrow, I'm not sure. So, satisfiability problems -- it turns out that there are cases that -- there are problems in the real world that end up being expressed essentially as lists of constraints where you're trying to find some, say, assignment of values to variables that satisfy the constraints. So an example might be scheduling people to work shifts in a hospital, right? Filling out the nurse shifts in a hospital. Different people have different constraints, some don't want to work at night, no individual can work more than this many hours out of that many hours, these two people don't want to be on the same shift, you have to have at least this many per shift and so on. So you can often describe a setting like that as a bunch of constraints on a set of variables. There's an interesting application of satisfiability that's going on here at MIT in the Lab for Computer Science, in fact I want to put a link to Daniel Jackson's home page, maybe you can help me to remember to do that. So Professor Jackson's doing this thing where he's interested in trying to find bugs in programs. So that's a good thing to do, but he wants to get the computer to do it automatically. And one way to do it is to essentially make a small example instance of a program. So an example of a kind of program that he might want to try to find a bug in would be an air traffic controller. So there's -- the air traffic controller has all these rules about how it works, right?

Introduction to Satisfiability

Examples of Satisfiability Problems

Flow model of discourse

Chafe'76:

“Our data ... suggest that as a speaker moves from focus to focus (or from thought to thought) there are certain points at which they may be a more or less radical change in space, time, character configuration, event structure, or even world ... At points where all these change in a maximal way, an episode boundary is strongly present.”

Discourse Exhibits Structure!

- Discourse can be partitioned into segments, which can be connected in a limited number of ways
- Speakers use linguistic devices to make this structure explicit cue phrases, intonation, gesture
- Listeners comprehend discourse by recognizing this structure
 - Kintsch, 1974: experiments with recall
 - Haviland&Clark, 1974: reading time for given/new information

Segmentation: Agreement

Percent agreement — ratio between observed agreements and possible agreements

	A	B	C
=====	-	-	-
=====	+	-	-
=====	-	+	+
=====	-	-	-
=====	+	+	+
=====	-	-	-
=====	-	-	-

$$\frac{22}{8 * 3} = 91\%$$

Types of Structure

- Linear vs. hierarchical
 - Linear: paragraphs in a text
 - Hierarchical: chapters, sections, subsections



- Typed vs. untyped
 - Typed: introduction, related work, experiments, conclusions



Our focus: Linear segmentation

Results on Agreement

People can reliably predict segment boundaries!

Grosz&Hirschberg'92	newspaper text	74-95%
Hearst'93	expository text	80%
Passanneau&Litman'93	monologues	82-92%

Linguistic Basis: Lexical Cohesion

- Common assumption of unsupervised algorithms
 - Word repetition indicates topical cohesion [Halliday & Hasan, '76]
 - Variations in lexical distribution signal topic changes

What is the instantaneous **speed**?
Well, **speed** is not sign sensitive.

It's like a spacecraft in orbit or an elevator with a cut cable .

Example

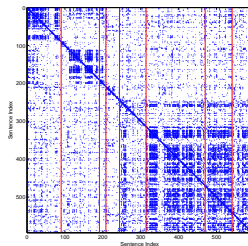
Stargazers Text(from Hearst, 1994)

- Intro - the search for life in space
- The moon's chemical composition
- How early proximity of the moon shaped it
- How the moon helped life evolve on earth
- Improbability of the earth-moon system

DotPlot Representation

Key assumption: change in lexical distribution signals topic change (Hearst '94)

- Dotplot Representation: (i, j) – similarity between sentence i and sentence j



Example

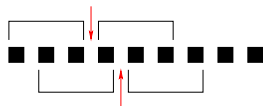
Sentence:	05	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95
14 form	1	111	1	1						1	1	1	1	1	1	1	1	1	1
8 scientist				11				1	1						1	1	1	1	1
5 space	11	1	1												1				
25 star	1			1								11	22	111112	1	1	1	11	1111
5 binary												11	1		1				1
4 trinary												1	1		1				1
8 astronomer	1			1								1	1		1	1	1	1	1
7 orbit	1				1							12	1	1	1				
6 pull						2	1	1						1	1				
16 planet	1	1		11					1					21	11111			1	1
7 galaxy	1															1	11	1	1
4 lunar				1	1	1	1	1											
19 life	1	1	1						1	11	1	11	1	1			1	1	111
27 moon			13	1111	1	1	22	21	21	21		11	1						
3 move										1	1	1							
7 continent										2	1	2	1						
3 shoreline												12							
6 time					1					1	1	1	1						1
3 water										11									
6 say								1	1			11							
3 species										1	1	1							

Outline

- Local similarity-based algorithm
- Global similarity-based algorithm
- HMM-based segmentor

Segmentation Algorithm of Hearst

- Initial segmentation
 - Divide a text into equal blocks of k words
- Similarity Computation
 - compute similarity between m blocks on the right and the left of the candidate boundary



- Boundary Detection
 - place a boundary where similarity score reaches local minimum

Similarity Computation: Representation

Vector-Space Representation

SENTENCE₁: I like apples

SENTENCE₂: Apples are good for you

Vocabulary	Apples	Are	For	Good	I	Like	you
Sentence ₁	1	0	0	0	1	1	0
Sentence ₂	1	1	1	1	0	0	1

Similarity Computation: Cosine Measure

Cosine of angle between two vectors in n-dimensional space

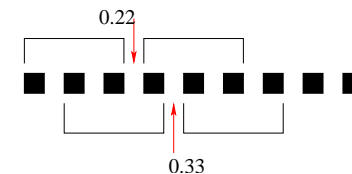
$$\text{sim}(b_1, b_2) = \frac{\sum_t w_{y,b_1} w_{t,b_2}}{\sqrt{\sum_t w_{t,b_1}^2 \sum_{t=1}^n w_{t,b_2}^2}}$$

SENTENCE₁: 1 0 0 0 1 1 0

SENTENCE₂: 1 1 1 1 0 0 1

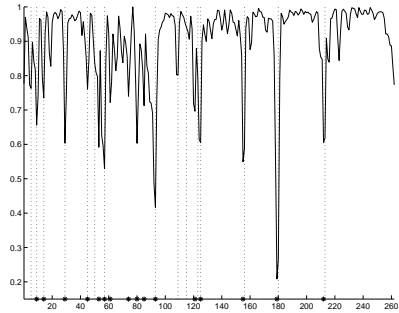
$$\text{sim}(S_1, S_2) = \frac{1*0+0*1+0*1+0*1+1*0+1*0+0*1}{\sqrt{(1^2+0^2+0^2+0^2+1^2+1^2+0^2)*(1^2+1^2+1^2+1^2+0^2+0^2+1^2)}} = 0.26$$

Output of Similarity computation:



Boundary Detection

- Boundaries correspond to local minima in the gap plot

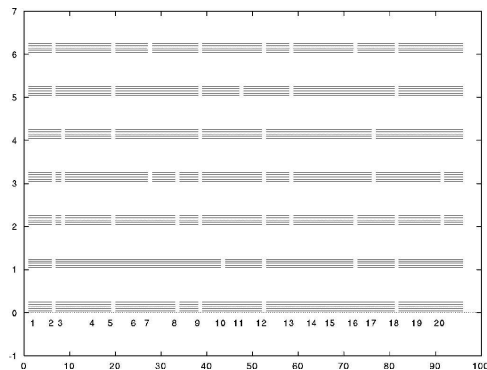


- Number of segments is based on the minima threshold ($s - \sigma/2$, where s and σ correspond to average and standard deviation of local minima)

Segmentation Evaluation

Comparison with human-annotated segments(Hearst'94):

- 13 articles (1800 and 2500 words)
- 7 judges
- boundary if three judges agree on the same segmentation point



Evaluation Results

Methods	Precision	Recall
Random Baseline 33%	0.44	0.37
Random Baseline 41%	0.43	0.42
Original method+thesaurus-based similarity	0.64	0.58
Original method	0.66	0.61
Judges	0.81	0.71

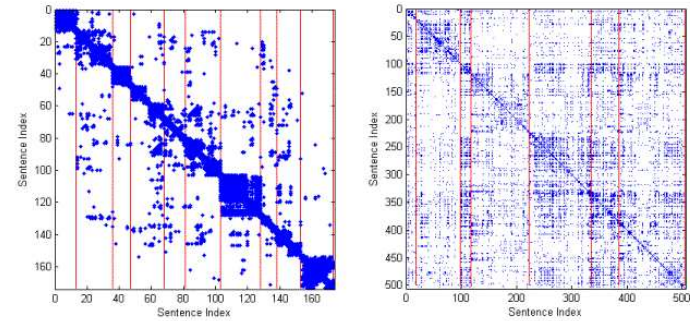
More Results

- High sensitivity to changes in parameter values
 - Parameters: Block size, window size and boundary threshold
- Thesaural information does not help
 - Thesaurus is used to compute similarity between sentences
 - synonyms are considered to be identical
- Most of the mistakes are “close misses”

Outline

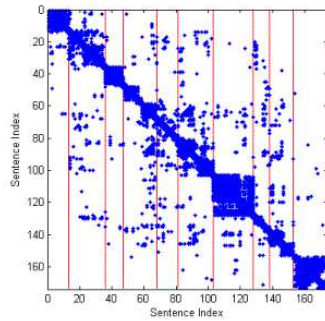
- Local similarity-based algorithm
- Global similarity-based algorithm
- HMM-based segmentor

Synthetic vs. Real Data



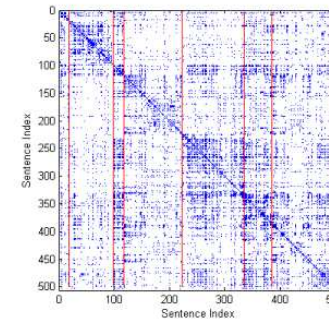
Synthetic Text Dotplot

- Broadcast News, synthetic document collections
- Exhibit sharp segment transitions



Physics Lecture Dotplot

- Spoken Lecture Data
- Exhibit very subtle topical transitions



Motion → Instantaneous Velocity → Average Acceleration → Numerical Example

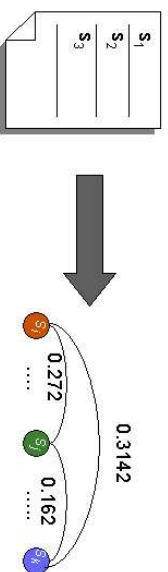
Minimum Cut Segmentation

- New graph-theoretic formalization of the segmentation objective
 - jointly maximizes within-cluster similarity and minimizes between-cluster similarity
 - Incorporates long-range lexical dependencies
- Exact, fast decoding using dynamic programming

Key Strength: Can detect subtle topic changes

Graph Based Representation

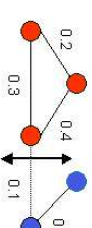
- Let $G(V,E)$ be a weighted, undirected, fully-connected graph
- Graph nodes represent textual units (e.g. sentences)
- Edge weights $w(s_i, s_j)$ indicate pairwise sentence similarity



Graph Cut Definitions

- Graph Cut - partitioning of the graph into two disjoint sets of nodes A, B
- **Between-segment similarity** (Cut Value) - sum of the edge weights between A, B
- **Within-segment similarity** (Volume) - sum of the edge weights for nodes in A
- Normalized Cut Value [Shi & Malik '00]:

$$Ncut(A, B) = \frac{cut(A, B)}{vol(A)} + \frac{cut(A, B)}{vol(B)}$$



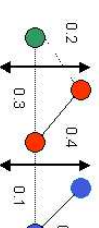
Normalized Cut Value = 0.6

Multi-way Graph Cuts

- K-way Graph Cut: partitioning of the graph into K disjoint sets, A_1, \dots, A_k
- K-way Normalized Cut Value:

$$Ncut_k(A_1, \dots, A_k) = \frac{cut(A_1, V - A_1)}{vol(A_1)} + \dots + \frac{cut(A_k, V - A_k)}{vol(A_k)}$$

- 3-way Cut Example:



Normalized Cut Value = 1.8

Optimization Objective

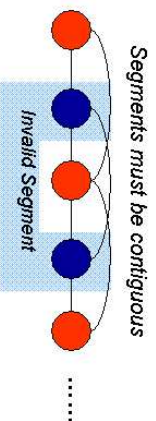
- For given k , we seek the k -way cut that minimizes the normalized cut value:

$$\min_{A_1, \dots, A_k} \frac{cut(A_1, V - A_1)}{vol(A_1)} + \dots + \frac{cut(A_k, V - A_k)}{vol(A_k)}$$

- With this objective, we jointly
 - minimize the Cut Value \sim similarity between segments
 - maximize the Volume \sim similarity within segments

Linearity Constraint

- Without further constraints, this optimization is NP -complete [Papadimitriou '00]
- However, the segmentation problem imposes a natural **linearity** constraint on the form of the solution:



- Reformulation: $\min_{A_1, \dots, A_k} Ncut_k(A_1, \dots, A_k)$
s.t. linearity constraint

Dynamic Programming Solution

Exact solution can be found using dynamic programming in $O(kn^2)$ time:

- $C[i, m]$: Minimum normalized cut of the segmentation of the first m sentences into l segments

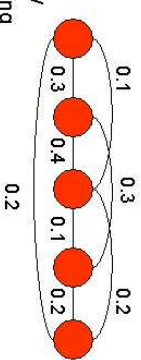
- $C[i, m]$ can be computed recursively by choosing the best sentence j prior to m to begin the l th segment:

$$C[i, m] = \min_{j < m} \left[C[i - 1, j] + \frac{cut[A_{j,m}, V - A_{j,m}]}{vol[A_{j,m}]} \right]$$

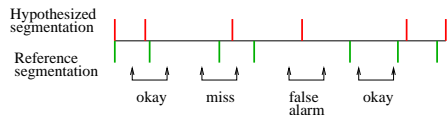
Graph Construction

- Node representation
 - Fixed blocks of text
- Topology
 - Fully-connected Graph
- Edge Weights
 - Weighted Cosine Similarity
 - Word Occurrence Smoothing

$$\tilde{s}_i = \sum_{j=i}^{i+k} e^{-\alpha(j-i)} s_j,$$



Evaluation Metric: P_k Measure



P_k : Probability that a randomly chosen pair of words k words apart is inconsistently classified (Beeferman '99)

- Set k to half of average segment length
- At each location, determine whether the two ends of the probe are in the same or different location. Increase a counter if the algorithm's segmentation disagree
- Normalize the count between 0 and 1 based on the number of measurements taken

Notes on P_k measure

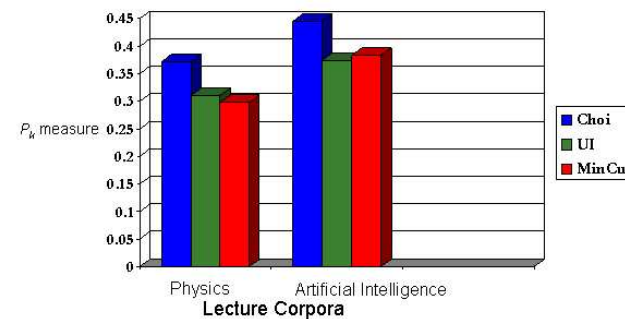
- $P_k \in [0, 1]$, the lower the better
- Random segmentation: $P_k \approx 0.5$
- On synthetic corpus: $P_k \in [0.05, 0.2]$
- On real segmentation tasks: $P_k \in [0.2, 0.4]$

Experiments

- Data: MIT Physics and AI Lecture Corpus
 - Verbose and colloquial language
 - Subtle topic transitions
 - Automatic Speech Recognition Error
- Baselines: State-of-the-art unsupervised segmentation systems
 - Utiyama & Isahara (UI) 2001 - language modeling approach
 - Choi 2000 - local similarity-based approach

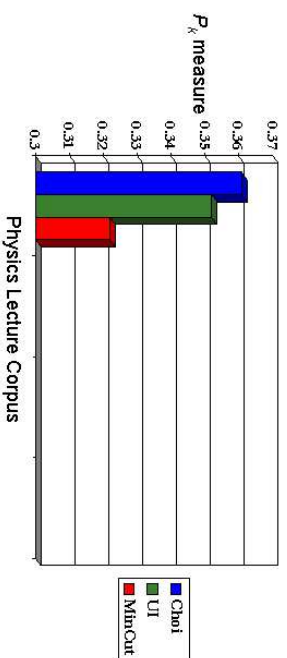
To control for segmentation granularity, the target number of segments for the baselines and our system is fixed

Results: Manually Transcribed Data



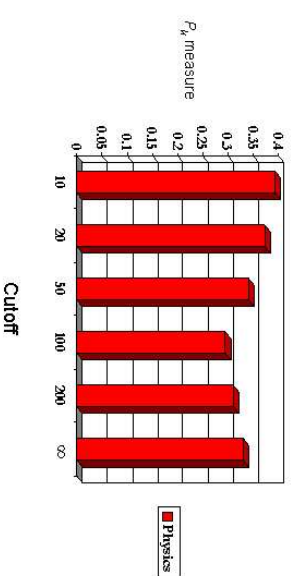
Human Agreement: $P_k \in [0.22; 0.42]$

Results: ASR Data (WER = 20%)



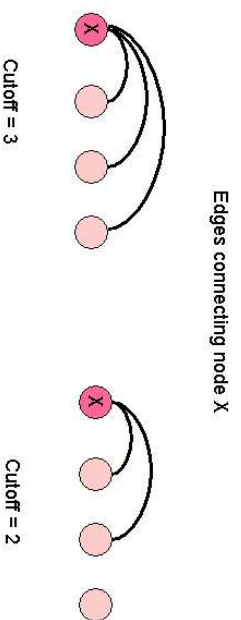
Impact of Long-range Dependencies

- Long-range dependencies improve performance



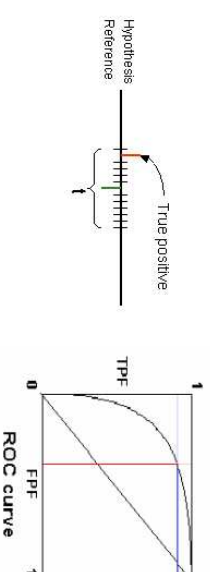
Impact of Long-range Dependencies

- Experiment: remove edges between nodes separated by a specified cutoff

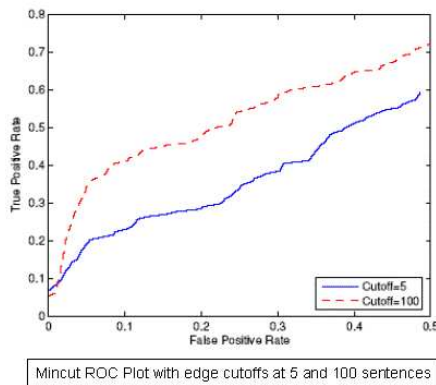


Evaluation Metrics - ROC

- Receiver Operating Characteristic Curve – represents tradeoff between true positives and false positives
- In segmentation, a true positive is a hypothesized boundary that occurs within a threshold t of the true boundary
- By varying t , we obtain points along the ROC curve



ROC Plot: Physics Lecture Data



Outline

- Local similarity-based algorithm
- Global similarity-based algorithm
- HMM-based segmentor

Typed Segmentation

- Task: determining the positions at which topics change in a stream of text or speech and **identify the type of each segment**.
- Example: divide newsstream into stories about sports, politics, entertainment, etc. Story boundaries are not provided. List of possible topics is provided.
- Straightforward solution: use a segmentor to find story boundaries and then assign to each story a topic label.
 - Segmentation mistakes may interfere with the classification step.
 - Combining the two steps can increase the accuracy

Types of Constraints

- “Local”: *negotiations* is more likely to predict the topic **politics** rather than **entertainment**
- “Contextual”: **politics** is more likely to start the broadcast than to follow **sports**



Hidden Markov Models for Segmentation

- We have a text with sentences s_1, s_2, \dots, s_n
(s_i is the i th sentence in the text)
 - $s_i = \{w_{i,1}, w_{i,2}, \dots, w_{i,m}\}$
- We have a topic sequence $T = t_1, t_2, \dots, t_n$
- We'll use an HMM to define

$$P(t_1, t_2, \dots, t_n, s_1, s_2, \dots, s_n)$$

for any text and tag sequence of the same length

- The most likely tag sequence for a text is

$$T^* = \operatorname{argmax}_T P(T, S)$$

- Topic breaks occur if $t_i \neq t_{i+1}$

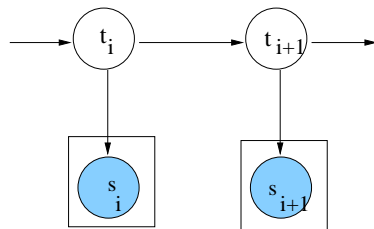
Hidden Markov Models for Segmentation

$$\begin{aligned} P(T, S) &= P(\text{END} | t_1, t_2, \dots, t_n, s_1, s_2, \dots, s_n) \times \\ &\prod_{j=1}^n [P(t_j | s_1, \dots, s_{j-1}, t_1, \dots, t_{j-1}) \times P(s_j | s_1, \dots, s_{j-1}, t_1, \dots, t_{j-1}, t_j)] \\ &= P(\text{END} | t_n) \times \prod_{j=1}^n [P(t_j | t_{j-1}) \times P(s_j | t_j)] \end{aligned}$$

Assumptions:

- Each topic t_i depends only on previous topic t_{i-1}
- Each sentence s_i only depends on topic t_i that generates it

Hidden Markov Models for Segmentation



- Choose a topic from an initial distribution of topics
- Generate a sentence from a distribution of words associated with a topic
- Choose another topic, possibly the same topic from a distribution of allowed transitions
- Repeat the process

Training the Models

- Fully supervised:
 - A newstream where stories are segmented and annotated with their type
- Partially supervised:
 - A newstream where stories are segmented but without type annotation
 - * During the preprocessing, cluster the stories based on cosine similarity or other distributional similarity metric

Parameter Estimation and Decoding

- Emission probabilities are modeled using a smoothed unigram model
(stop words are removed during preprocessing)

$$P(s|t) = \prod_i (w_i|t)$$

- Transition probabilities are based on ML estimates

$$P(\text{sports}|\text{politics}) = \frac{\text{count}(\text{sports}, \text{politics})}{\text{count}(\text{politics})}$$

- Using Viterbi algorithm, recover a tag sequence for a given sequence of sentences

Results

- Evaluation data: 2.2 million words (6,000 stories) from CNN and ABC
The data is transcribed automatically
- Evaluation measures:
 - P_{Miss} — probability of missed boundary (within window of 50 words)
 - $P_{FalseAlarm}$ — probability of false segmentation (within window of 50 words)
 - $C_{Seg} = P_{Seg} * P_{Miss} + (1 - P_{Seg}) * P_{FalseAlarm}$, where P_{Seg} is the *a priori* probability of a segment boundary being within the window length ($P_{Seg} = 0.3$)
- Results:

Show	P_{Miss}	$P_{FalseAlarm}$	P_{Seg}
ABC	0.3453	0.088	0.158
CNN	0.3094	0.1022	0.164