Tokenization

Markus Dickinson
Corpora & Linguistic Knowledge
April 10, 2002
dickinso@ling.ohio-state.edu

Tokenization

- Background/Motivation:
  - 1. Where tokenization fits into annotation tools
  - 2. What the task depends on
  - 3. Concerns (Abbreviations)
- Approaches:
  - 4. "Informed" approach
  - 5. Statistical approach
  - 6. Hybrid approach

1. How tokenization fits into NLP

- Preprocessing > Tokenization > Morphological analysis (e.g. POS tagging)
- Tokenization = dividing the input text into tokens
  - words, which can have further morphological analysis and belong to a certain syntactic class
  - character(s) with recognizable structure, e.g. punctuation, numbers, dates
- Closely linked to the task of sentence segmentation (sentence-final vs. abbreviatory periods)

(Pre-)Classification

- Tokenization can also be done as labeled tokenization or as a step to pipeline into classification.
- e.g number, date, time, title classifications – pipelines in the LT TTT program. Tokenize a number differently because you've identified it as a number, so pass that information along.
- Can be tricky:
  - *Dr. North* vs. *Oak Dr. North*
  - *They sold 1996 {bales of hay / cars}.*

Preprocessing

- Filter out typesetting distinctions
- However, font information can be important (Grefenstette 1999) -- can use markup information
- Some tasks -- e.g. dehyphenation -- can be seen either as preprocessing or as tokenization proper

2. 4 Dependencies (Palmer)

- Language Dependence
- Character-set Dependence
- Application Dependence
- Corpus Dependence

Language Dependence

- Spoken or written
- Logographic, syllabic, or alphabetic
- Sentence/word marking: Amharic texts explicitly mark word & sentence boundaries; Thai marks neither; English is in-between.
  - English is a space-delimited language; Chinese is an unsegmented language
- Many tasks, e.g. regular expression matching, will be language-specific

Character-set Dependence

- Asciifiation: *de'ja'* or *de1ja2* (normally, a word ending in a number might be an abbreviation)
- Multiple encodings exist for the same character set -- e.g. GB & Big-5 for Chinese -- but tokenize the characters the same
- byte range 161-191 are punctuation marks in Latin-1 encoding of English; the same range are Thai consonants in TIS620 encoding
  - might be code-switching in the text

Application Dependence

- No absolute definition for tokenization
- Contraction expansion: might want to expand *I'm* into *I am* if we want to parse later (*I'm* might be an unknown word).
- Proper names like *John Jones* could be one token for most purposes, but if tokenization is a preprocessing step for a family identification program, not a good idea
- *ACL* is 3 phonological words, one orthographic abbreviation (Sproat et al 1996)

Corpus Dependence

- *governor's* is expanded/split in the Susanne corpus (two tokens); one token in the Brown corpus
- cannot expect a corpus to follow set conventions on spelling, punctuation, etc. (e-mail text will likely be "ill-formed")
- LOB Corpus uses \0 to signal a one-word abbreviation (Leech 1997 in Garside et al 1997): *\0in.* (abbreviation) vs. *in.* (sentence-final)

3. Concerns

- Recoverability -- eliminating space/tab/newline distinction (for languages with spaces)
- very langauge-specific tasks:
  - Same or different?: *3.9 to 4 million dollars* vs. *$3.9 to $4 million*
  - Multipart words: *Boston-based dogs, = dog + ,*
  - Multiword expressions: *in spite of* (≈*despite*)
  - Abbreviations

Abbreviations

- Abbreviations are productive: cannot produce a list of all of them
- Abbreviations can also be words: *mass* is also the short form of *Massachusetts*
- Abbreviations can represent different words & so be in different contexts: *St.* = *Saint*, *State*, or *Street*. (*Saint* less likely to be at a sentence boundary)

Abbreviations (cont.)

- Contraction Expansion: expand *I'm* into *I am*
  - need to know that *'m* = *am*
  - probably would also want to expand non-apostrophe-containing words like Spanish *del* = *de* + *el*.
- Punctuation can be very ambiguous
  - for an in-depth discussion of punctuation, as it applies to linguistics in general, see Nunberg 1990

Punctuation

- Apostrophes: *I'll* -- one token or two?
- Hyphens: line-final because initially one seamless word or initially hyphenated?
- Periods: sentence-final or abbreviatory?

Apostrophes

- If we split *I'll* into *I* and *'ll* we now have "funny words" (Manning & Schutze 1999) in the data.
- If we do not split, then rules like S -> NP VP no longer apply for sentences like *I'm right*.
- Apostrophes can be used as single quotes, so there is potential for ambiguity.
- As mentioned before, could be used in something like *de'ja'*
- Word-internal uses: *Pudd'n'head rock 'n' roll*

Hyphens

- If we dehyphenate (for the line-final cases), it is possible that the original corpus information may be lost -- can probably add markup.
- Dehyphenating all line-final cases will over-dehyphenate.
- Some hyphenated words are one word: *e-mail*, *co-operate*, *non-lawyer*
- Some are not one word: *text-based*, *sound-change*

Periods

- The bulk of research has concerned separating sentence-final periods from ones denoting abbreviations.
  - *The bizarre 12 in. alien told us to come in.*
  - note that the analysis would have repercussions for a text-to-speech system ([IntSIz] vs. [In])
- Ties into the area of sentence segmentation

Haplology

- One character/string having 2 simultaneous uses
- Apply your tokenizer to the following "corpus":
  - *"'Whose frisbee is this?' John asked, rather self-consciously. 'Oh, it's one of the boys' said the Sen."*
- The hyphen, apostophe/single quote, & final period are all serving two uses.
- Usually, we want to handle the two uses differently -- e.g. split a sentence-final period from the preceding word (unless an abbreviation).

Approaches

- Now that we've identified the problems of tokenization, we can examine some solutions
- (Linguistically) Informed Approach
- Statistical Approach
- Hybrid Approach

4. Informed Approaches

- use something about the language to find abbrevs. (Grefenstette & Tapanainen 1994, Grefenstette1999):
  - match regular expressions (for segmented lgs.)
  - use a corpus filter
  - use a lexicon
  - use a list of abbreviations
- similar techniques for unsegmented languages
- Use as a baseline: any period not followed by a blank is not a full stop; otherwise, a full stop.

Regular Expressions

- can use a tool like lex/flex or awk
- (English) numbers: ([0-9]+[,])*[0-9]([.][0-9]+)?
- single capital letters: [A-Za-z]\.
- *L.L.*: [A-Za-z]\.([A-Za-z0-9]\.)+
- capital letter + consonants + period (eg. *Assn.*): [A-Z][bcdfghj-np-tvxz]+\.
- Will not catch: *Gen. 25-ft. USN.*
- possible to use other regular expressions
- will not work with unsegmented languages

Corpus filter

- Identify likely abbreviations, ending with a period & followed by:
  - another piece of punctuation, a lower-case letter, a number, or a word beginning with a capital letter & ending in a period.
- Then, use the corpus as a filter: if the likely abbreviations appears elsewhere in the corpus without a period, remove it from the likely list
  - note, however, that *OH.* & *OH* can both appear in a corpus (although, not likely)

Use a lexicon

- First, identify numbers & separate on spaces
- Ordered filter (part of the morphological analyzer):
  - 1. Define: **known abbreviation**: followed by lowercase letter, comma, or semi-colon
  - 2. Prune: lowercase, exists in the lexicon w/o a period --> not an abbreviation;
  - 2. Add: lowercase otherwise (always with a period): abbreviation

Use a lexicon (cont.)

  - 3. Prune: begins w/ uppercase letter, is not a known abbreviation, appears elsewhere w/o a period; or appears only once or twice --> not an abbreviation (probably proper name)
  - 4. Else: an abbreviation
- still some problems: *in. (=inch)* as an abbreviation will be ruled out by all the other non-abbreviatory uses

Use abbreviations in the lexicon

- Simply define abbreviations in the lexicon that:
  - a) are fairly hard to detect otherwise
  - b) do not exist as words otherwise (e.g. *in.* would wrongly identify sentence-ending prepositions)
- New procedure:
  - 1. abbreviation = followed by lowercase letter, comma, or semi-colon
  - 2. abbreviation = exists as abbreviation in lexicon
  - 3. otherwise, sentence terminator
  - could probably also use some of previous pruning

Unsegmented Languages

- Need a more informed approach: extensive word list & an informed segmentation algorithm
- unknown words are difficult
- Approaches
  - character-as-word
  - greedy algorithm for word-matching
- Native speakers disagree: Sproat et al (1996) report Chinese speakers agree on word segmentation around 70% of the time

Character-as-word

- Average word length in Chinese is 1-2 characters
- Not good for parsing, POS tagging, text-to-speech
- good for information retrieval systems
- also, not very general: same type of strategy wouldn’t work for an alphabetic system

Greedy algorithm

- Or maximum matching algorithm
- Start at the first character & match the longest word in the word list starting with that character
  - if matches a word, mark the end of the longest word & start with the next character after that word
  - if doesn’t match, segment that character as a word & begin again at the next character
- variation: match a sequence of unmatched characters
- ignores ambiguity -- one segmentation

Forward-backward matching

- English: *thetabledownthere* would be segmented as *theta bled own there*
- If we match back-to-front, however, we obtain *the table down there* (reverse maximum matching)
- Forward-backward matching: compare results of forward matching with reverse matching (use language-specific heuristics)

5. Statistical approach(es)

- Kiss & Strunk 2002 treat period-ending abbreviations as bigrams
- count up the occurrences of a word with a period & occurrences without in a training corpus
- use the log likelihood ratio to determine if the bigram is a true collocation
- Compare the null hypothesis that the period is independent of the preceding word with the alternative hypothesis that it is not independent.
  - a similar idea to Grefenstette's corpus filter

Log likelihood ratio

- Null hypothesis: H_0: $P(\bullet, w) = p = P(\sim\bullet, w)$
  - in other words, the probability of a word occurring with a period is independent of whether it occurs elsewhere without a period (i.e. not a collocation)
- Alternative: H_Λ: $P(\bullet, w)\ p_1 \neq p_2\ P(\sim\bullet, w)$
  - the occurrence of a period is not independent (either because it is most likely a collocation or most likely not a collocation)
- $\log \lambda = -2 \log (L(H_0)/L(H_\Lambda))$
  - where L(X) is the likelihood of X -- calculated using the probabilities of a collocation, themselves based on the occurrence counts

Scaling the log likelihood ratio

- The method gets most of the abbreviations, but generates many false positives -- e.g. says *holiday* is an abbreviation
- Using C(*word*,•) as the count of *word* with a following period and C(*word*,~•) as the count of *word* without a following period, we scale by:
  - ratio of occurrence: $e^{C(word,\bullet)/C(word,\sim\bullet)}$
  - relative difference: $(C(word,\bullet)-C(word,\sim\bullet))/(C(word,\bullet)+C(word,\sim\bullet))$
  - length of abbreviation: $1/(e^{length})$

6. Hybrid approaches

- Some approaches use lexical-based knowledge & then use statistical information to pick out the best segmentation from a set of choices
- e.g. could use POS information to rank the choices
  - note that this breaks down the straight-line flow of information from tokenizer to morphological analyzer & POS tagger
- Sproat et al follow a similar route of selecting a best choice

Chinese word segmentation

- Sproat et al 1996
- use a dictionary -- ideally from the same genre as the text to be separated.
- "coverage of the dictionary … [is] possibly more important than the particular set of methods used in the segmentation"
  - e.g. *huang2-rong2 you1-you1 de dao4* 'Huang Rong soberly said' where *you1-you1* = 'soberly' A different system attached *you1* to the preceding name because its dictionary lacked *you1-you1*.

Main Problems

- Unknown words:
  - Morphologically derived words: *xue2-sheng1*+**men0** = 'student' + plural = 'students'
  - Personal names: *shi2-ji1-lin2* will not be in any dictionary
  - Transliterated foreign words: *bu4-lang3-shi4-wei2-ke4* = 'Brunswick'

Why hybrid?

- Words in the lexicon treated with the FST (next slide). Unknown words treated by deriving them via productive (linguistic) processes
- e.g. morphological analysis for some affixes
- e.g. use information like the semantic radical to estimate the probability of a sequence of characters being a name
  - radical: some characters have general semantic meanings, like GHOST or GOLD

Finite-State Transducer

- use a weighted finite-state transducer
  - each arc corresponds to a Chinese character (or to epsilon) and has a numerical weight assigned to it (obtained from a training corpus)
  - the cheapest path through the FST will be the chosen segmentation
- e.g. ABCD could be ABC / D or AB / CD
  - The ABC path costs 6.0 & D costs 5.0, so path = 11.0
  - AB costs 4.0 & CD costs 5.0, so path = 9.0
  - ergo, choose AB / CD

Finite-State Transducer (cont.)

- All characters have zero weight; the actual weighting takes place on final POS arcs, which denote that the previous arcs made up a word:

Limitations

- Will only catch local ambiguities
- if the ambiguity resolution depends on broader context, the WFST cannot handle it
- e.g. *ma3-lu4* means either 'horse way' or 'road' depending on the global context of the sentence
  - *The **horse** got sick on the **way**.* ('this CL horse way on sick ASP')
  - *Very few cars pass by this **road**.* ('this CL road very few car pass by')

LT TTT

- Also hybrid: some components are rule-based, others are statistical
- Documentation found at:
  http://www.ltg.ed.ac.uk/software/ttt/tttdoc.html
  or
  file:/opt/compling/tools/TTT_v1.0/DOC/tttdoc.html
- Different components do different tasks, like number/date/time identifcation

“Assignment”

- >cd /opt/compling/tools/TTT_v1.0
- >setenv TTT /home/compling/tools/TTT_v1.0
- >more runplain
  - # read the comments to get an idea of what the pipelines are doing
- >cat $TTT/EGS/plain/texts | $TTT/runplain > /home/<username>/texts.html
- open up netscape & load the file texts.html
- try running one pipeline at a time to see what each component is doing

References

- Garside, Roger, Leech, Geoffrey, and McEnery, Tony (Eds.) (1997). *Corpus Annotation: linguistic information from computer text corpora*. Harlow, England: Addison Wesley Longman Limited
- Grefenstette, Gregory (1999). Tokenization. In *Syntactic Wordclass Tagging*, H. van Halteren, ed., pp. 117-133. Dordrecht: Kluwer Academic Publishers.
- Grefenstette, Gregory and Tapanainen, Pasi (1994). What is a word, What is a sentence? Problems of tokenization. In *Third Conference on Computational Lexicography and Text Research (COMPLEX-94)*. Budapest, Hungary. http://www.xrce.xerox.com/publis/mltt/mltt-004.ps.
- Kiss, Tibor, and Strunk, Jan (2002). Scaled log likelihood ratios for the detection of abbreviations in text corpora. ms.
- Manning, Christopher D., and Schutze, Hinrich (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

References (cont.)

- Nunberg, Geoffrey. 1990. *The Linguistics of Punctuation*. Stanford, CA: Center for the Study of Language and Information.
- Palmer, David D. (2000). Tokenisation and Sentence Segmentation. In Dale, Robert, Moisl, Herman, and Somers, Harold, eds. *Handbook of Natural Language Processing*, pp. 11-35. New York: Marcel Dekker. http://www.netLibrary.com/ebook_info.asp?product_id=47610.
- Sproat R., Shih C., Gale W., Chang N. 1996. A Stochastic Finite-State Word-Segmentation Algorithm for Chinese. *Computational Linguistics*, 22(3).