CIS 5300: NATURAL LANGUAGE PROCESSING

Word Tokenization

Professor Chris Callison-Burch

Text Normalization

Every NLP task needs to do text normalization:

1. Segmenting/tokenizing words in running text
2. Normalizing word formats
3. Segmenting sentences in running text

How Many Words?

I do uh main- mainly business data processing

- Fragments, filled pauses

Seuss's cat in the hat is different from other cats!

- **Lemma**: same stem, part of speech, rough word sense
  - cat and cats = same lemma
- **Wordform**: the full inflected surface form
  - cat and cats = different wordforms

How Many Words?

they lay back on the San Francisco grass and looked at the stars and their

Type: an element of the vocabulary.

Token: an instance of that type in running text.

How many?

- 15 tokens (or 14)
- 13 types (or 12) (or 11?)

How Many Words?

N = number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

The relationship between the number of words in the vocabulary as the size of the corpus grows is given by

Gale Church 1990

$$\boxed{|V| = kN^{\beta}}$$

$$|V| > O(N^{1/2})$$

Simple Tokenization in UNIX

(Inspired by Ken Church's UNIX for Poets.)

Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt
     | sort
     | uniq –c
     | sort -nr
```

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

Sort numerically descending

The first step: tokenizing

```
tr -sc 'A-Za-z' '\n' < shakes.txt | head
```

```
THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...
```

The second step: sorting

```
tr -sc 'A-Za-z' '\n' < shakes.txt | sort | head
```

```
A
A
A
A
A
A
A
A
A
...
```

More counting

Merging upper and lower case

```
tr 'A-Z' 'a-z' < shakes.txt | tr –sc 'A-Za-z' '\n' | sort | uniq –c
```

Sorting the counts

```
tr 'A-Z' 'a-z' < shakes.txt | tr –sc 'A-Za-z' '\n' | sort | uniq –c | sort –n –r
```

```
23243 the
22225 i
18618 and
16339 to
15687 of
12780 a
12163 you
10839 my
10005 in
8954  d
```

What happened here?

called
ca ll'd

killed
kill'd

Issues in Tokenization

Finland's capital	→	Finland Finlands Finland's ?
what're, I'm, isn't	→	What are, I am, is not
Hewlett-Packard	→	Hewlett Packard ?
state-of-the-art	→	state of the art ?
Lowercase	→	lower-case lowercase lower case ?
San Francisco	→	one token or two?
m.p.h., PhD.	→	??

Tokenization: language issues

French

- ***L'ensemble*** → one token or two?
  - ***L*** ? ***L'*** ? ***Le*** ?
  - Want ***l'ensemble*** to match with ***un ensemble***

German noun compounds are not segmented

- ***Lebensversicherungsgesellschaftsangestellter***
- 'life insurance company employee'
- German information retrieval needs **compound splitter**

Tokenization: language issues

Chinese and Japanese no spaces between words:

- 莎拉波娃现在居住在美国东南部的佛罗里达。
- 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达

Further complicated in Japanese, with multiple alphabets intermingled

- Dates/amounts in multiple formats

フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

Katakana Hiragana Kanji Romaji

End-user can express query entirely in hiragana!

Word Tokenization in Chinese

Also called **Word Segmentation**

Chinese words are composed of characters

- Characters are generally 1 syllable and 1 morpheme.
- Average word is 2.4 characters long.

Standard baseline segmentation algorithm:

- Maximum Matching (also called Greedy)

Maximum Matching Word Segmentation Algorithm

Given a wordlist of Chinese, and a string:

1) Start a pointer at the beginning of the string
2) Find the longest word in dictionary that matches the string starting at pointer
3) Move the pointer over the word in string
4) Go to 2

Max-match segmentation illustration

Thecatinthehat

Thetabledownthere

the cat in the hat

the table down there

theta bled own there

Doesn’t generally work in English!

But works surprisingly well in Chinese

- 莎拉波娃现在居住在美国东南部的佛罗里达。
- 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达

Modern probabilistic segmentation algorithms even better

CIS 5300: NATURAL LANGUAGE PROCESSING

Word Normalization and Stemming

Professor Chris Callison-Burch

Normalization

Need to “normalize” terms

- Information Retrieval: indexed text & query terms must have same form.
  - We want to match ***U.S.A.*** and ***USA***

We implicitly define equivalence classes of terms

- e.g., deleting periods in a term

Alternative: asymmetric expansion:

- Enter: ***window*** Search: ***window, windows***
- Enter: ***windows*** Search: ***Windows, windows, window***
- Enter: ***Windows*** Search: ***Windows***

Potentially more powerful, but less efficient

Case folding

Applications like IR: reduce all letters to lower case

- Since users tend to use lower case
- Possible exception: upper case in mid-sentence?
  - e.g., ***General Motors***
  - ***Fed*** vs. ***fed***
  - ***SAIL*** vs. ***sail***

For sentiment analysis, MT, Information extraction

- Case is helpful (***US*** versus ***us*** is important)

Lemmatization

Reduce inflections or variant forms to base form

- *am, are, is* $\rightarrow$ *be*
- *car, cars, car's, cars'* $\rightarrow$ *car*

the boy's cars are different colors $\rightarrow$ *the boy car be different color*

Lemmatization: have to find correct dictionary headword form

Machine translation

- Spanish quiero (‘I want’), quieres (‘you want’) same lemma as querer ‘want’

Morphology

Morphemes:

- The small meaningful units that make up words
- **Stems**: The core meaning-bearing units
- **Affixes**: Bits and pieces that adhere to stems
- Often with grammatical functions

Stemming

Reduce terms to their stems in information retrieval

Stemming is crude chopping of affixes

- language dependent
- e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

→

for exampl compress and compress ar both accept as equival to compress

Porter's algorithm: the most common English stemmer

Step 1a

```
sses → ss   caresses → caress
ies  → i    ponies   → poni
ss   → ss   caress   → caress
s    → ø    cats     → cat
```

Step 1b

```
(*v*)ing → ø  walking   → walk
              sing      → sing
(*v*)ed  → ø  plastered → plaster
…
```

Step 2 (for long stems)

```
ational→ ate relational→ relate
izer→ ize    digitizer → digitize
ator→ ate    operator  → operate
…
```

Step 3 (for longer stems)

```
al   → ø  revival    → reviv
able → ø  adjustable → adjust
ate  → ø  activate   → activ
…
```

Viewing morphology in a corpus

Why only strip –ing if there is a vowel?

$$(*v*)\text{ing} \rightarrow \emptyset \quad \texttt{walking} \rightarrow \texttt{walk}$$
$$\texttt{sing} \rightarrow \texttt{sing}$$

Viewing morphology in a corpus

Why only strip –ing if there is a vowel?

```
(*v*)ing → ø   walking   → walk
               sing      → sing
```

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep 'ing$' | sort | uniq -c | sort –nr
```

```
1312 King
 548 being
 541 nothing
 388 king
 375 bring
 358 thing
 307 ring
 152 something
 145 coming
 130 morning
```

```
548 being
541 nothing
152 something
145 coming
130 morning
122 having
120 living
117 loving
116 Being
102 going
```

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep '[aeiou].*ing$' | sort | uniq -c | sort –nr
```

Dealing with complex morphology is sometimes necessary

Some languages requires complex morpheme segmentation

- Turkish
- Uygarlastiramadiklarimizdanmissinizcasina
- `(behaving) as if you are among those whom we could not civilize’

- Uygar `civilized’ + las `become’
  + tir `cause’ + ama `not able’
  + dik `past’ + lar ‘plural’
  + imiz ‘p1pl’ + dan ‘abl’
  + mis ‘past’ + siniz ‘2pl’ + casina ‘as if’

CIS 5300: NATURAL LANGUAGE PROCESSING

Sentence Segmentation and Decision Trees

Professor Chris Callison-Burch

Sentence Segmentation

!, ? are relatively unambiguous

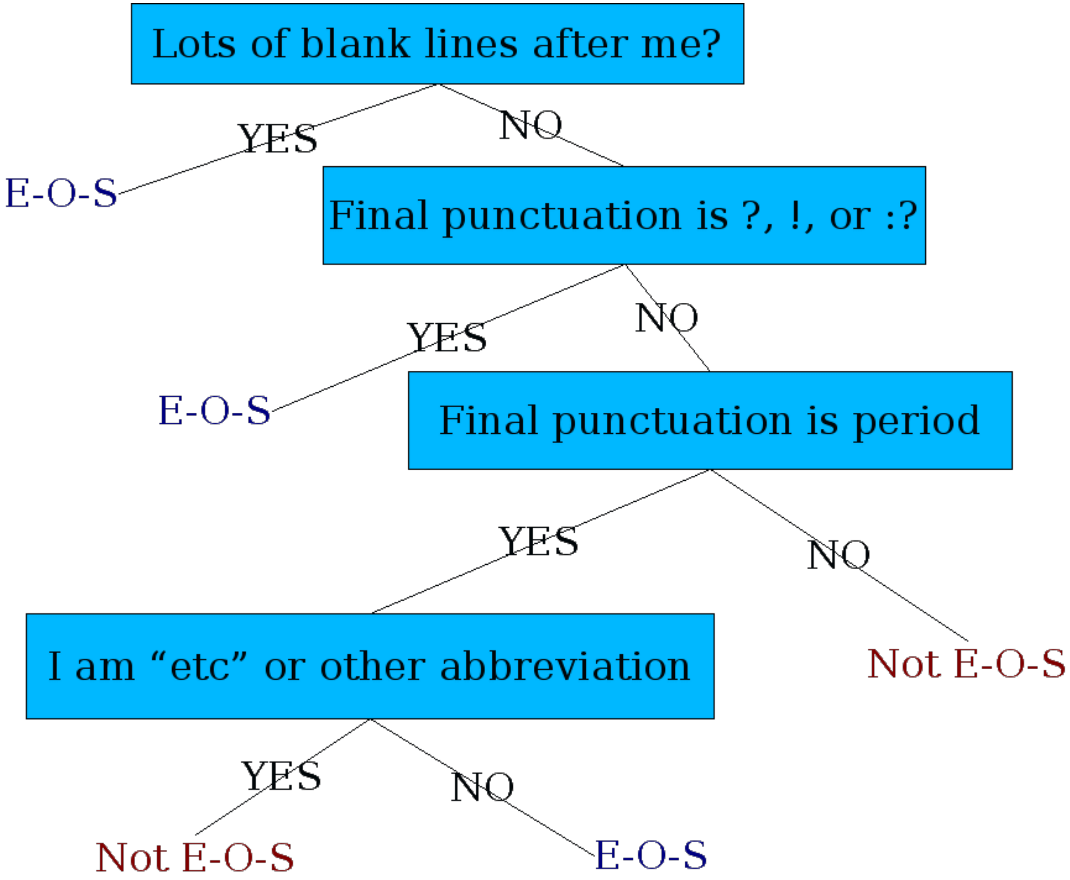
Period "." is quite ambiguous

- Sentence boundary
- Abbreviations like Inc. or Dr.
- Numbers like .02% or 4.3
- URLs www.google.com

Build a binary classifier

- Looks at a "."
- Decides EndOfSentence/NotEndOfSentence
- Classifiers: hand-written rules, regular expressions, or machine-learning

Determining if a word is end-of-sentence: a Decision Tree



More sophisticated decision tree features

Case of word with “.”: Upper, Lower, Cap, Number

Case of word after “.”: Upper, Lower, Cap, Number

Numeric features

- Length of word with “.”
- Probability (word with “.” occurs at end-of-s)
- Probability (word after “.” occurs at beginning-of-s)

Implementing Decision Trees

A decision tree is just an if-then-else statement

The interesting research is choosing the features

Setting up the structure is often too hard to do by hand

- Hand-building only possible for very simple features, domains
    - For numeric features, it’s too hard to pick each threshold
- Instead, structure usually learned by machine learning from a training corpus

Decision Trees and other classifiers

We can think of the questions in a decision tree

As features that could be exploited by any kind of classifier

- Logistic regression
- SVM
- Neural Nets
- etc.