Text Processing

Word Tokenization

Tokenization is the process of segmenting a string of characters into tokens (words).

A swimmer likes swimming, thus he swims.

↓

a	swimmer	likes	swimming	thus	he	swims

An example

I have a can opener; but I can't open these cans.

- Word Tokens: 11
- Word Types: 10

Several tokenization libraries

- NLTK Toolkit (Python)
- Spacy (Python)
- Polyglot (Python)
- Stanford CoreNLP (Java)
- Unix Commands

Issues in Tokenization

Common examples

- Finland's → Finland Finlands Finland's ?
- What're, I'm, shouldn't → What are, I am, should not ?
- San Francisco → one token or two?
- m.p.h. → ??

Hyphenation

- **End-of-Line Hyphen**: Used for splitting whole words into part for text justification. e.g. *"... apparently, mid-dle English followed this practice..."*
- **Lexical Hyphen**: Certain prefixes are offen written hyphenated, e.g. *co-, pre-, meta-, multi-, etc.*
- **Sententially Determined Hyphenation**: Mainly to prevent incorrect parsing of the phrase. e.g. *State-of-the-art, three-to-five-year, etc.*

Language Specific Issues

French

l’ensemble: want to match with un ensemble

Language Specific Issues

French

l'ensemble: want to match with un ensemble

German

Noun coumpounds are not segmented

- Lebensversicherungsgesellschaftsangestellter
- 'life insurance company employee'

Language Specific Issues

French

l'ensemble: want to match with un ensemble

German

Noun coumpounds are not segmented

- Lebensversicherungsgesellschaftsangestellter
- 'life insurance company employee'

Sanskrit

Very long compound words

सत्यम्ब्रूयात्प्रियम्ब्रूयान्नब्रूयात्सत्यमप्रियम्प्रियश्चनानृतम्ब्रूयादेषधर्मःसनातनः

Language Specific Issues

Chinese

No space 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

Sharapova now lives in US southeastern Florida

Japanese

Further complications with multiple alphabets intermingled.

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

Longest Words

Max	Language (non scientific)
431	**Sanskrit** *(Longest)*
173	Greek
136	Afrikaans
85	Māori
79	German
74	Turkish
64	Icelandic
56	Hungarian
54	Spanish
49	Dutch
46	Malay
45	English
44	Romanian
42	Georgian
41	Czech
39	Bulgarian
39	Lithuanian
36	Kazakh
33	Norwegian
32	Tagalog
32	Polish
30	Serbian
30	Montenegrin
30	Italian
30	Croatian

Word Tokenization in Chinese

Maximum Matching (Greedy Algorithm)

- Start a pointer at the beginning of the string
- Find the largest word in dictionary that matches the string starting at pointer
- Move the pointer over the word in string

Will the above scheme work for English?

Word Tokenization in Chinese

Maximum Matching (Greedy Algorithm)

- Start a pointer at the beginning of the string
- Find the largest word in dictionary that matches the string starting at pointer
- Move the pointer over the word in string

Will the above scheme work for English?

No: Thetabledownthere

Yes: #ThankYouSachin, #musicmonday etc.

Text Segmentation for Sanskrit

[1]

General assumption behind the design

Sentences from Classical Sanskrit may be generated by a regular relation R of the Kleene closure W^* of a regular set W of *words* over a finite alphabet Σ.

- W: vocabulary of (inflected) words (*padas*) and
- R: sandhi

Analysis of a sentence

A candidate sentence w is analyzed by inverting relation R to produce a finite sequence $w_1, w_2, \ldots w_n$ of word forms, together with a proof that $w \in R(w_1 \cdot w_2 \ldots \cdot w_n)$.

[1] http://sanskrit.inria.fr

Sentence Segmentation

Can we decide where the sentences begin and end?

Why it is difficult?

- Are ‘!’ and ‘?’ ambiguous?

Sentence Segmentation

Can we decide where the sentences begin and end?

Why it is difficult?

- Are ‘!’ and ‘?’ ambiguous? No Is period “.”
- ambiguous?

Sentence Segmentation

Can we decide where the sentences begin and end?

Why it is difficult?

- Are '!' and '?' ambiguous? No Is period "."
- ambiguous? Yes
  - Abbreviations (Dr., Mr., m.p.h.)

Sentence Segmentation

Can we decide where the sentences begin and end?

Why it is difficult?

- Are ‘!’ and ‘?’ ambiguous? No Is period “.”
- ambiguous? Yes
  - Abbreviations (Dr., Mr., m.p.h.)
  - Numbers (2.4%, 4.3)

Sentence Segmentation

Can we decide where the sentences begin and end?

Why it is difficult?

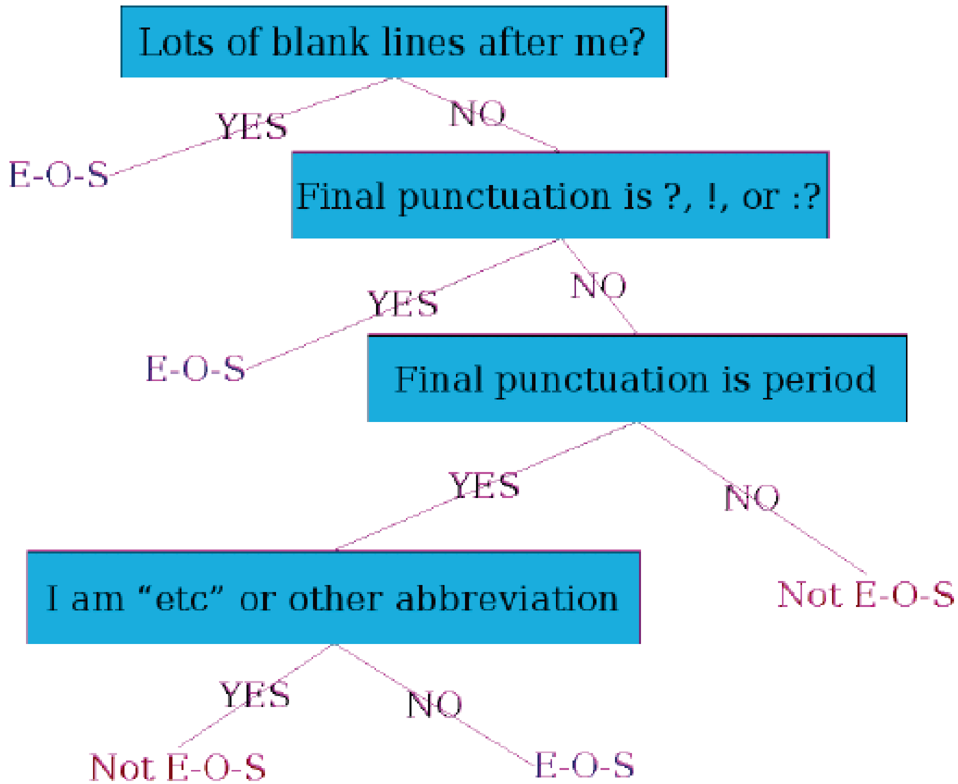
- Are '!' and '?' ambiguous? No
- Is period "." ambiguous? Yes
  - Abbreviations (Dr., Mr., m.p.h.)
  - Numbers (2.4%, 4.3)

Can we build a binary classifier for 'period' classification? For each "."

- Decides EndOfSentence/NotEndOfSentence
- Classifiers can be: hand-written rules, regular expressions, or machine learning

Sentence Segmentation: Decision Tree Example

Decision Tree: Is this word the end-of-sentence (E-O-S)?



Other Important Features

- Case of word with ".": Upper, Lower, Number
- Case of word after ".": Upper, Lower, Number
- Numeric Features
  - Length of word with "."
  - Probability (word with "." occurs at end-of-sentence)
  - Probability (word after "." occurs at beginning-of-sentence)

Implementing Decision Trees

- Just an if-then-else statement
- Choosing the features is more important
- For numeric features, thresholds are to be picked
- With increasing features including numerical ones, difficult to set up the structure by hand
- Decision Tree structure can be learned using machine learning over a training corpus

Implementing Decision Trees

- Just an if-then-else statement
- Choosing the features is more important
- For numeric features, thresholds are to be picked
- With increasing features including numerical ones, difficult to set up the structure by hand
- Decision Tree structure can be learned using machine learning over a training corpus

Basic Idea

Usually works top-down, by choosing a variable at each step that best splits the set of items.

Popular algorithms: ID3, C4.5, CART

Other Classifiers

- Support Vector Machines
- Logistic regression
- Neural Networks

Normalization

Why to "normalize"?

Indexed text and query terms must have the same form.

- U.S.A. and USA should be matched
- We implicitly define equivalence classes of terms

Case Folding

- Reduce all letters to lower case
- Some caveats (Task dependent):
    - Upper case in mid sentence, may point to named entities (e.g. General Motors)
    - For MT and information extraction, some cases might be helpful (*US* vs. *us*)

Python tokenization example

```
import nltk
text = "This is Andrew's text, isn't it?"
```

```
tokenizer = nltk.tokenize.WhitespaceTokenizer()
tokenizer.tokenize(text)
```

```
['This', 'is', "Andrew's", 'text,', "isn't", 'it?']
```

```
tokenizer = nltk.tokenize.TreebankWordTokenizer()
tokenizer.tokenize(text)
```

```
['This', 'is', 'Andrew', "'s", 'text', ',', 'is', "n't",
'it', '?']
```

```
tokenizer = nltk.tokenize.WordPunctTokenizer()
tokenizer.tokenize(text)
```

```
['This', 'is', 'Andrew', "'", 's', 'text', ',', 'isn',
 "'", 't', 'it', '?']
```

http://text-processing.com/demo/tokenize/

Simple Tokenization in UNIX

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

```
tr -sc 'A-Za-z' '\n' < file_name
| sort
| uniq -c
| sort -rn
```

- Change all non-alphabetic characters to newline
- Sort in alphabetical order
- Merge and count each type
- Sort based on the count

For more info: execute **'man tr'**

Token normalization

We may want the same token for different forms of the word

- wolf, wolves → wolf
- talk, talks → talk

Stemming

- A process of removing and replacing suffixes to get to the root form of the word, which is called the **stem**
- Usually refers to heuristics that chop off suffixes

Lemmatization

- Usually refers to doing things properly with the use of a vocabulary and morphological analysis
- Returns the base or dictionary form of a word, which is known as the **lemma**

Lemmatization example

WordNet lemmatizer

- Uses the WordNet Database to lookup lemmas
- nltk.stem.WordNetLemmatizer
- Examples:
  - feet → foot
  - wolves → wolf

cats → cat
talked → talked

- Problems: not all forms are reduced

- Takeaway: we need to try stemming or lemmatization and choose best for our task

Lemmatization

- Reduce inflections or variant forms to base form:
  - am, are, is → be
  - car, cars, car's, cars' → car
- Have to find the correct dictionary headword form

Lemmatization in Python

```
>>> from nltk.stem import WordNetLemmatizer
>>> wordnet_lemmatizer = WordNetLemmatizer()
>>> wordnet_lemmatizer.lemmatize('dogs')
u'dog'
>>> wordnet_lemmatizer.lemmatize('churches')
u'church'
>>> wordnet_lemmatizer.lemmatize('abaci')
u'abacus'
```

Morphology

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

Morphology

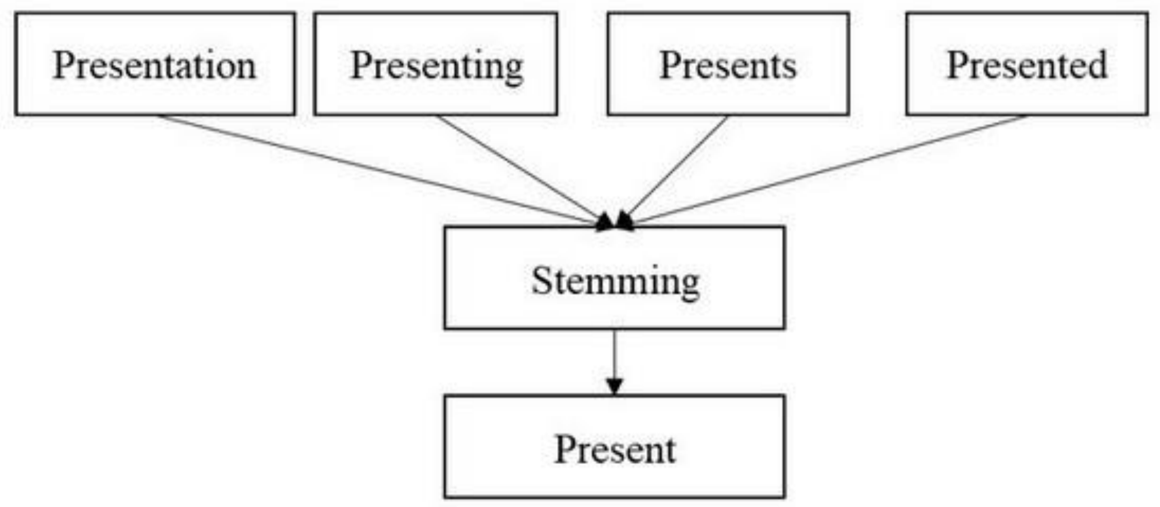
Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

Morphemes are divided into two categories

- Stems: The core meaning bearing units
- Affixes: Bits and pieces adhering to stems to change their meanings and grammatical functions
  - Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
  - Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)

Stemming

- Reducing terms to their stems
- Used in information retrieval Crude chopping of affixes
- Language dependent



Porter's algorithm

Step 1a

- sses → ss (caresses → caress)
- ies → i (ponies → poni)
- ss → ss (caress → caress)
- s → ϕ(cats → cat)

Step 1b

- (*v*)ing → ϕ(walking → walk, king →

Porter's algorithm

Step 1a

- sses → ss (caresses → caress)
- ies → i (ponies → poni)
- ss → ss (caress → caress)
- s → ϕ(cats → cat)

Step 1b

- (*v*)ing → ϕ(walking → walk, king → king)
- (*v*)ed → ϕ(played → play)
- ...

If first two rules of Step 1b are successful, the following is

- done: AT → ATE (conflat(ed) → conflate)
- BL → BLE (troubl(ed) → trouble)
-

Porter's algorithm

Step 2

- ational → ate (relational → relate)
- izer → ize (digitizer → digitize)
- ator → ate (operator → operate)
- ...

Porter's algorithm

Step 2

- ational → ate (relational → relate)
- izer → ize (digitizer → digitize)
- ator → ate (operator → operate)
- ...

Step 3

- al → ϕ(revival → reviv)
- able → ϕ(adjustable → adjust)
- ate → ϕ(activate → activ)
- ...

Complete Algorithm is available at:
http://snowball.tartarus.org/algorithms/porter/stemmer.html

Python stemming example

```
import nltk
text = "feet cats wolves talked"
tokenizer = nltk.tokenize.TreebankWordTokenizer()
tokens = tokenizer.tokenize(text)
```

```
stemmer = nltk.stem.PorterStemmer()
" ".join(stemmer.stem(token) for token in tokens)
```

```
u'feet cat wolv talk'
```

```
stemmer = nltk.stem.WordNetLemmatizer()
" ".join(stemmer.lemmatize(token) for token in tokens)
```

```
u'foot cat wolf talked'
```

Stemming in Python

```
>>> from nltk.stem.porter import PorterStemmer
>>> porter_stemmer = PorterStemmer()
>>> porter_stemmer.stem('maximum')
'maximum'
>>> porter_stemmer.stem('presumably')
'presum'
>>> porter_stemmer.stem('multiply')
'multipli'
>>> porter_stemmer.stem('provision')
'provis'
```