

Regular expressions: docs

- python: <https://docs.python.org/2/library/re.html>
- java: <http://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html>

Basic Text Processing

Word Normalization and Stemming

Klinton Bicknell

(borrowing from: Dan Jurafsky and Jim Martin)

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* → *be*
 - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *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 equal 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 → ∅	cats → cat

Step 1b

(*v*)ing → ∅	walking → walk
	sing → sing
(*v*)ed → ∅	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 → ∅	revival → reviv
able → ∅	adjustable → adjust
ate → ∅	activate → activ
...	

Viewing morphology in a corpus

Why only strip -ing if there is a vowel?

(*v*)ing → ∅ walking → walk
 sing → sing

Viewing morphology in a corpus

Why only strip -ing if there is a vowel?

(*v*)ing → ∅ walking → walk
 sing → sing

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

1312 King	548 being
548 being	541 nothing
541 nothing	152 something
388 king	145 coming
375 bring	130 morning
358 thing	122 having
307 ring	120 living
152 something	117 loving
145 coming	116 Being
130 morning	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’

Basic Text Processing

Word Normalization and
Stemming

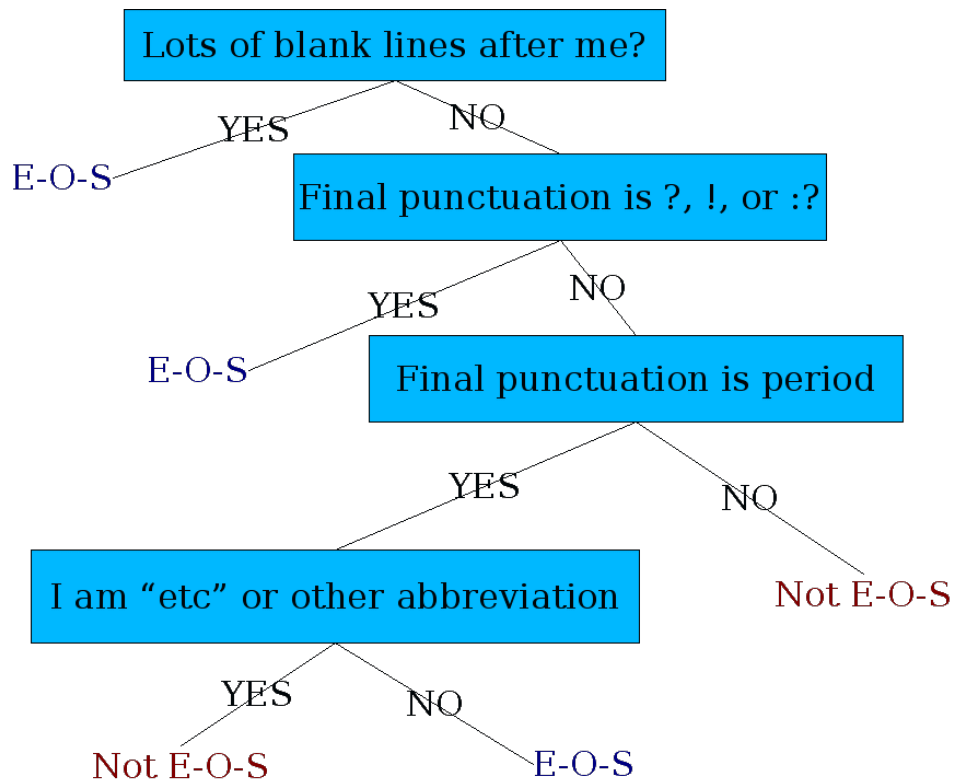
Basic Text Processing

Sentence Segmentation
and Decision Trees

Sentence Segmentation

- !, ? are relatively unambiguous
- Period “.” is quite ambiguous
 - Sentence boundary
 - Abbreviations like Inc. or Dr.
 - Numbers like .02% or 4.3
- 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.

Basic Text Processing

Sentence Segmentation
and Decision Trees