Basic NLP Text Processing

MIE223
Winter 2025

1 Basic NLP Text Processing. NLP=Natural Language Processing

1.1 NLP Text Processing Pipeline

nltk provides implementations for most operations

- Document $\rightarrow$ Sections and Paragraphs
- Paragraphs $\rightarrow$ Sentences (sentence segmentation / extraction)
- Sentences $\rightarrow$ Tokens
- Tokens $\rightarrow$ Lemmas or Morphological Variants / Stems
- Tokens $\rightarrow$ Part-of-speech (POS) Tags
- Tokens, POS Tags $\rightarrow$ Phrase Chunks (Noun & Verb Phrases)
- Tokens, POS Tags $\rightarrow$ Parse Trees
  - Augment above with coreference, entailment, sentiment, ...

2 Basic Text Processing: Word tokenization

2.1 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

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

$$N = \text{number of tokens} \tag{1}$$
$$V = \text{vocabulary} = \text{set of types} \tag{2}$$
$$|V| = \text{size of vocabulary} \tag{3}$$
$$|V| > O(N^{1/2}) \tag{4}$$

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

2.2 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. → ??

Keep hyphens together? Remove contractions? Does upper and lower case matter?

2.3 Tokenization: language issues

Languages can have different tokenization issues such as French and German. French has 'l'ensemble' which can be one token or two. German has long noun compounds that are not segmented. Chinese and Japanese have no spaces between words. Japanese also has different scripts that can be used in the same sentence.

3 Word Normalization and Stemming

3.1 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

3.2 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)

3.3 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'

3.4 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

3.5 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. $\rightarrow$ for exampl compress and compress ar both accept as equival to compress

3.6 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

…

3.7 Viewing morphology in a corpus Why only strip –ing if there is a vowel?

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

```
(*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
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

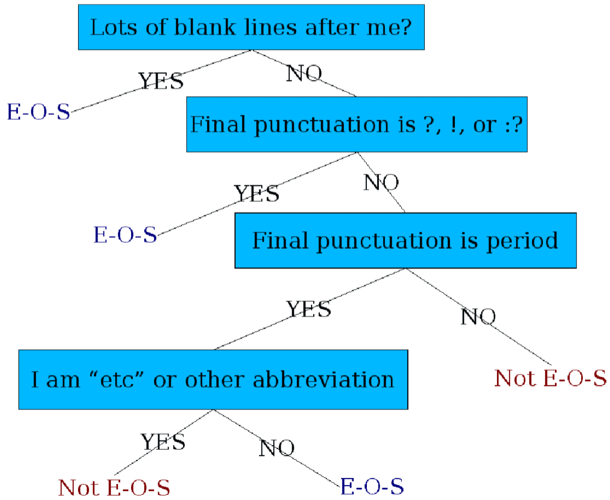
```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep '[aeiou].*ing$' | sort | uniq -c | sort -nr
```

4 Sentence Segmentation and Decision Trees

4.1 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
- f(input) = True, False
- "The car is traveling 10 m.p.h."
- Check which period is ending the sentence
  - Looks at a "."
  - Decides EndOfSentence/NotEndOfSentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning

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



4.3 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)

5 Regular Expressions: Detecting word pattern variations

5.1 Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks

5.2 Regular Expressions: Disjunctions

Letters inside square brackets []

Pattern	Matches
[wW]oodchuck	Woodchuck, woodchuck
[1234567890]	Any digit

Ranges [A-Z]

Pattern	Matches	
[A-Z]	An upper case letter	Drenched Blossoms
[a-z]	A lower case letter	my beans were impatient
[0-9]	A single digit	Chapter 1: Down the Rabbit Hole

5.3 Regular Expressions: Negation in Disjunction

Negations [Ŝs]
Carat means negation only when first in []

Pattern	Matches	
[^A-Z]	Not an upper case letter	Oyfn pripetchik
[^Ss]	Neither 'S' nor 's'	I have no exquisite reason"
[^e^]	Neither e nor ^	Look here
a^b	The pattern a carat b	Look up a^b now

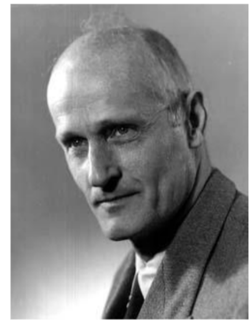
5.4 Regular Expressions: More Disjunction

Woodchucks is another name for groundhog! The pipe — for disjunction.

Pattern	Alternative
groundhog|woodchuck	
yours|mine	
a|b|c	= [abc]
[gG]roundhog|[Ww]oodchuck	

5.5 Regular Expressions: ? * + .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n



Stephen C Kleene

Kleene *, Kleene +

5.6 Regular Expressions: Anchors: ˆ$

Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Za-z]	1 "Hello"
\.$	The end.
.$	The end? The end!

5.7 Example

Find me all instances of the word "the" in a text.
the: Misses capitalized examples
[tT]he: Incorrectly returns other or theology
[â-zA-Z][tT]he[â-zA-Z]

5.8 Errors

- The process we just went through was based on fixing two kinds of errors
  - Matching strings that we should not have matched (there, then, other)
    - False positives (Type I)
  - Not matching things that we should have matched (The)
    - False negatives (Type II)

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
  - Increasing accuracy or precision (minimizing false positives)
  - Increasing coverage or recall (minimizing false negatives).

5.9 Regex Summary

- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers
  - Can be very useful in capturing generalizations