https://xkcd.com/208/

CS 1671/2071
Human Language Technologies

Session 2: Text normalization

Michael Miller Yoder

January 13, 2025

School of Computing and Information

About Norah (TA)

- **2nd year PhD student at Pitt.**

- **Office Hours:**
  - By appointment.
  - Email: nia135@pitt.edu
  - Location TBD

Hacking 4 Humanity hackathon

- Tech and policy hackathon
- Last year, a team from here at SCI won 1st in the technology track and was invited to Harrisburg to present their projects to to First Lady Lori Shapiro and members of Governor Shapiro's staff
- More information at www.hacking4humanity.online/

Overview: Text normalization

- Course logistics
- Basic terminology
- Regular expressions
- Text normalization

Course logistics

- Reading for today was Jurafsky & Martin sections 2-2.3, 2.5-2.7
- Assignments and quizzes are generally due Thu nights in this course
- Homework 1 has been released. Is due next Thu Jan 23
- Please remind me of your name before asking or answering a question (just this class session)

NLP terminology: words and corpora

How many words in this phrase?

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

- How many?
  - 15 tokens (or 14 if you count "San Francisco" as one)
  - 13 types (or 12) (or 11?)
- **Type**: a unique word in the vocabulary
- **Token**: an instance of a word type in running text
- **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

Slide adapted from Jurafsky & Martin

How many words in a corpus?

Corpus: a (machine-readable) collection of texts

N = number of tokens

V = vocabulary = set of types, |*V*| is size of vocabulary

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13+ million

Slide adapted from Jurafsky & Martin

Corpora vary along dimensions like

- Texts don't appear out of nowhere!
- Language: 7097 languages in the world
- Variety, like African American Language varieties.
  - AAE Twitter posts might include forms like "*iont" (I don't)*
- Code switching, e.g., Spanish/English, Hindi/English:

  Por primera vez veo a @username actually being helpful! It was beautiful:)

  [For the first time I get to see @username actually being helpful! it was beautiful:)]

  dost tha or ra- hega ... dont wory ... but dherya rakhe

  [“he was and will remain a friend ... don’t worry ... but have faith”]

- Genre: newswire, fiction, scientific articles, Wikipedia
- Author Demographics: writer's age, gender, ethnicity, SES
- Corpus datasheets [Bender & Friedman 2018, Gebru+ 2020] ask about this information

Slide adapted from Jurafsky & Martin

Regular expressions (regex)

Regular expressions

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

Slide adapted from Jurafsky & Martin

Regular expressions: Disjunctions (OR)

- Letters inside square brackets []

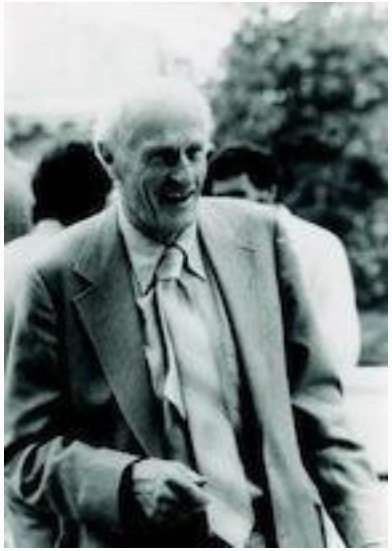
Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit

- Ranges `[A-Z]` `[a-z]` `[0-9]`
- Negations `[^A-Z]`
  - Carat means negation only when first in []
- Sequence disjunctions with pipe |
  - `groundhog|woodchuck`

Slide adapted from Jurafsky & Martin

Regular expression wildcards: *+.

Pattern	Matches	
`oo*h`	0 or more of previous char	oh ooh oooh ooooh
`o+h`	1 or more of previous char	oh ooh oooh ooooh
`beg.n`	Any char	begin begun begun beg3n



Stephen C Kleene

Slide adapted from Jurafsky & Martin

Regular expression example

- Find all instances of the word “the” in a text.

```
the
```

- Misses capitalized examples

```
[tT]he
```

- Incorrectly returns "other" or "theology"

```
[^a-zA-Z][tT]he[^a-zA-Z]
```

Slide adapted from Jurafsky & Martin

Errors

The process we just went through was based on fixing two kinds of errors:

1. Matching strings that we should not have matched (there, then, other)

   False positives (Type I errors)

2. Not matching things that we should have matched (The)

   False negatives (Type II errors)

Slide adapted from Jurafsky & Martin

Simple Application: ELIZA

- Early NLP system that imitated a Rogerian psychotherapist [Weizenbaum 1966]
- Uses pattern matching to match phrases

  `“I need X”`

- and translates them into, e.g.

  `“What would it mean to you if you got X?`

Slide adapted from Jurafsky & Martin

Simple Application: ELIZA

Men are all alike.
IN WHAT WAY

They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE

Well, my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE

He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED

Slide adapted from Jurafsky & Martin

How ELIZA works

```
.* I'M (depressed|sad) .* → I AM SORRY TO HEAR YOU ARE \1
.* all .* → IN WHAT WAY?
.* always .* → CAN YOU THINK OF A SPECIFIC EXAMPLE?/
```

Slide adapted from Jurafsky & Martin

Regular expressions summary

- Regular expressions play a surprisingly large role in NLP
  - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For hard tasks, we use machine learning classifiers
  - But regular expressions are still used for pre-processing, or used to extract features for the classifiers

Slide adapted from Jurafsky & Martin

Text normalization (preprocessing)

Every NLP task requires text normalization

- Tokenizing (separating) words
- Normalizing word formats

Slide adapted from Jurafsky & Martin

Case folding (lowercasing)

- Applications like IR: reduce all letters to lowercase
  - Since users tend to use lowercase
  - 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)

Slide adapted from Jurafsky & Martin

Stopword removal

- Do we want to keep "function words" like *the*, *of*, *and*, *I*, *you*, etc?
- Sometimes no (information retrieval)
- Sometimes yes (authorship attribution)

Tokenization

Space-based tokenization

- A very simple way to tokenize
- For languages that use space characters between words
  - Arabic, Cyrillic, Greek, Latin, etc., based writing systems
- Segment off a token between instances of spaces

Slide adapted from Jurafsky & Martin

Issues in Tokenization

- Can't just blindly remove punctuation:
  - m.p.h., Ph.D., AT&T, cap’n
  - prices ($45.55)
  - dates (01/02/06)
  - URLs (http://www.pitt.edu)
  - hashtags (#nlproc)
  - email addresses (someone@cs.colorado.edu)
- Clitic: a word that doesn't stand on its own
  - "are" in we're, French "je" in j'ai, "le" in l'honneur
- When should multiword expressions (MWE) be words?
  - New York, rock ’n’ roll

Slide adapted from Jurafsky & Martin

Regex-based tokenization

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)    # set flag to allow verbose regexps
...     ([A-Z]\.)+         # abbreviations, e.g. U.S.A.
...   | \w+(-\w+)*         # words with optional internal hyphens
...   | \$?\d+(\.\d+)?%?  # currency and percentages, e.g. $12.40, 82%
...   | \.\.\.            # ellipsis
...   | [][.,;"'?():-_`]  # these are separate tokens; includes ], [
... '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

- NLTK [Bird+ 2009] provides regex and ML models for tokenization (like punkt tokenizer)
- spaCy, other packages provide good tokenization

Slide adapted from Jurafsky & Martin

Tokenization in languages without spaces between words

- Many languages (like Chinese, Japanese, Thai) don't use spaces to separate words!
- How do we decide where the token boundaries should be?

Slide adapted from Jurafsky & Martin

Word tokenization in Chinese

- Chinese words are composed of characters called "**hanzi**" (or sometimes just "**zi**")
- Each one represents a meaning unit called a morpheme
- Each word has on average 2.4 of them.
- But deciding what counts as a word is complex and not agreed upon.

Slide adapted from Jurafsky & Martin

How to do word tokenization in Chinese?

姚明进入总决赛 “Yao Ming reaches the finals”

3 words?
姚明 进入 总决赛
YaoMing reaches finals

5 words?
姚 明 进入 总 决赛
Yao Ming reaches overall finals

7 characters? (don't use words at all):
姚 明 进 入 总 决 赛
Yao Ming enter enter overall decision game

Slide adapted from Jurafsky & Martin

Word tokenization / segmentation

- In Chinese NLP it's common to just treat each character (zi) as a token.
  - So the segmentation step is very simple
- In other languages (like Thai and Japanese), more complex word segmentation is required, usually using machine learning

Slide adapted from Jurafsky & Martin

Lemmatization and stemming

Lemmatization

Represent words as their **lemma**: their shared root, dictionary headword form:

- *am, are, is* → *be*
- *car, cars, car's, cars'* → *car*
- Spanish quiero (‘I want’), quieres (‘you want’)

  → querer ‘want'
- *He is reading detective stories*

  → *He be read detective story*

Slide adapted from Jurafsky & Martin

Lemmatization is done by Morphological Parsing

- Morphemes: small meaningful units that make up words
  - Roots: The core meaning-bearing units
  - Affixes: Parts that adhere to roots

un-think-able; kitten-s

- Affixes can add grammatical meaning (inflections, 2nd column) or modify semantic meaning (derivations, 3rd column)

<root>	<root>ing	<root>er
run	running	runner
think	thinking	thinker
program	programming	programmer
kill	killing	killer

Slide adapted from Jurafsky & Martin, David Mortensen

Lemmatization is done by Morphological Parsing

- *cats* into two morphemes *cat* and *s*
- Spanish *amaren* (‘if in the future they would love’) into morpheme *amar* ‘to love’ + morphological features *3PL* + *future subjunctive*.

Slide adapted from Jurafsky & Martin

Dealing with complex morphology is necessary for many languages

- e.g., the Turkish word:

  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 '1pl' + dan 'abl'
  + mis 'past' + siniz '2pl' + casina 'as if'

Slide adapted from Jurafsky & Martin

Stemming

- Reduce terms to stems, chopping off affixes crudely

This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with

Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with

$$\begin{aligned} \text{ATIONAL} &\rightarrow \text{ATE} \quad \text{(e.g., relational} \rightarrow \text{relate)} \\ \text{ING} &\rightarrow \epsilon \quad \text{if stem contains vowel (e.g., motoring} \rightarrow \text{motor)} \\ \text{SSES} &\rightarrow \text{SS} \quad \text{(e.g., grasses} \rightarrow \text{grass)} \end{aligned}$$

Slide adapted from Jurafsky & Martin

Conclusion: Text normalization

- Regular expressions match flexible sequences of characters and allow substitution of groups of characters
- Stopwords are function words like “the”, “a”, “and”, “of”, etc that are often ignored in NLP applications
- Tokenization: splitting texts into sequences of words
    - Subword tokenization finds tokens based on frequencies of sequences of characters in data
- Lemmatization: normalizing words to their dictionary roots
- Stemming: chopping off affixes of words to reduce them to stems