https://xkcd.com/208/

CS 1671 / CS 2071 / ISSP 2071
Human Language Technologies

Session 4: Words, tokens and preprocessing

Michael Miller Yoder

January 26, 2026

University of Pittsburgh | School of Computing and Information

Course logistics: quiz

- First in-class quiz is **this Wed Jan 28**
  - Covers readings from all the sessions up to that point
  - Looking over the reading is a great way to prepare
  - Session 4: J+M 2-2.6, 2.8, 2.10
  - Can cover content assigned in reading that is not discussed in class
- Conceptual, not programming
- Lowest quiz score will be dropped
- Quizzes are (only) 15% of your course grade total

Course logistics: quiz

- On Canvas, 10 minutes to complete it (1-1:10pm)
- Allowed resources
  - Textbook
  - Your notes (on a computer or physical)
  - Course slides and website
- Resources not allowed
  - Generative AI
  - Internet searches

Course logistics

- Homework 1 will be released soon (tomorrow or Wed). Will be **due Feb 12**
- Please remind me of your name before asking or answering a question

PyLing

Pitt Python Linguistics Group

- Computational linguistics group on campus
- Practice NLP skills related to this class with fun tutorials and guest speakers (with food, too!)
- Next meeting is **Wed Jan 28, 6-7:15pm at CL 2818** (linguistics department conference room)
  - Meeting topic: text analysis of NLP conference proceedings (how much are dominated by LLMs? What other areas are prevalent?)
- Contact Na-Rae Han, naraehan@pitt.edu to get on their mailing list

Overview: Words, tokens and preprocessing

- Words and corpora
- Morphemes
- Tokenization and subword tokenization
- Regular expressions
- Other text preprocessing
- Coding activity: preprocessing Airbnb listings

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

Morphemes

Morphemes

- 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

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

Tokenization

Why tokenize?

- Using a deterministic series of tokens means systems can be compared equally
  - Systems agree on the length of a string
- Eliminates the problem of unknown words

Slide adapted from Jurafsky & Martin

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

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.
  - The standard algorithms are neural sequence models trained by supervised machine learning.

Slide adapted from Jurafsky & Martin

Subword tokenization & BPE

Another option for text tokenization

- **Use the data** to tell us how to tokenize.
- **Subword tokenization** (because tokens can be parts of words as well as whole words)
- Many modern neural NLP systems (like LLMs) use this to handle unknown words
- 2 parts:
  - A token learner that takes a raw training corpus and induces a vocabulary (a set of tokens).
  - A token segmenter that takes a raw test sentence and tokenizes it according to that vocabulary

Slide adapted from Jurafsky & Martin

Byte Pair Encoding (BPE) token learner

Iteratively merge frequent neighboring tokens to create longer tokens.

Start with all characters
Repeat:

- Choose most frequent neighboring pair ('A', 'B')
- Add a new merged symbol ('AB') to the vocabulary
- Replace every 'A' 'B' in the corpus with 'AB'.

Until *k* merges

Vocabulary

[A, B, C, D, E]

[A, B, C, D, E, AB]

[A, B, C, D, E, AB, CAB]

Corpus

A B D C A B E C A B

AB D C AB E C AB

AB D CAB E CAB

Slide adapted from Jurafsky & Martin

BPE token learner

Original (very fascinating 🙄) corpus:

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

Split on whitespace, add end-of-word tokens _

corpus		vocabulary
5	l o w _	_, d, e, i, l, n, o, r, s, t, w
2	l o w e s t _	
6	n e w e r _	
3	w i d e r _	
2	n e w _	

Slide adapted from Jurafsky & Martin

BPE token learner

- Merge e r to er

corpus		vocabulary
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er
2	l o w e s t _	
6	n e w er _	
3	w i d er _	
2	n e w _	

- Merge er _ to er_
- Merge n e to ne

Slide adapted from Jurafsky & Martin

BPE token learner

The next merges are:

Merge	Current Vocabulary
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

Slide adapted from Jurafsky & Martin

BPE token segmenter algorithm

- On the test data, run each merge learned from the training data:
  - Greedily, in the order we learned them
- So merge every e r to er, then merge er _ to er_, etc.
- Result:
  - Test set "n e w e r _" would be tokenized as a full word
  - Test set "l o w e r _" would be two tokens: "low er_"

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

Regular expressions: Disjunctions (OR)

- Letters inside square brackets []

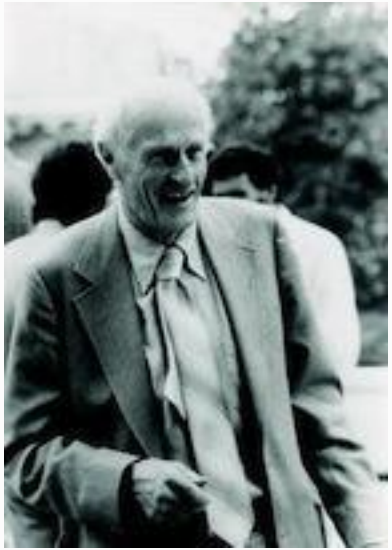
Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit

- 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

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

Other text preprocessing (normalization)

Case folding (lowercasing)

- Applications like information retrieval: 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

Lemmatization and stemming

Lemmatization: reducing words to their **lemmas**: 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*

Stemming: reducing words to their “stems”, chopping off affixes crudely. You aren’t left with true words, but is fast to run.

- *This was an accurate, complete copy of the map*
  → *Thi was an accur complet copi of the map*

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)

Conclusion: Words, tokens, and preprocessing

- Word types are unique words
- Morphemes are the smallest meaning-bearing units within words
- Unicode represent characters for many languages and scripts in code points, which can be encoded into bytes with UTF-8
- Tokenization: splitting texts into sequences of words
    - Subword tokenization finds tokens based on frequencies of sequences of characters in data
- Regular expressions match flexible sequences of characters
- Lemmatization: normalizing words to their dictionary roots
- Stopwords are function words like “the”, “a”, “and”, “of”, etc that are often ignored in NLP applications

Coding activity:
Preprocessing Airbnb listings

Load in-class notebook

1. Go to this nbgitpuller link (also available on course website)
2. Log in with your Pitt username if necessary
3. Start a server with **TEACH – 6 CPUs, 48 GB**
4. Load custom environment at **/ix1/cs1671-2026s/class_env**
5. This should pull the cs1671_spring2026_jupyterhub folder into your JupyterLab
6. Open **session4_preprocessing.ipynb**