Preprocessing Text

Natural Language Processing

Basic Preprocessing of Text

Text Preprocessing

Before we do most NLP, we want to be able to examine individual sentences and individual words.

It means we need to do:

- Sentence segmentation (sentence tokenization)
- Lexical analysis (word tokenization)

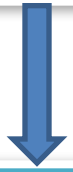
Sentence Segmentation

- The first step in sentence segmentation is sentence boundary detection.
- In English, what is the most obvious approach to that?

County sheriffs chased Alex Jones of San Jose all the way to Gilroy, where he lived. Jones had run from a crime scene near his employer. The sheriff's department promised more details by tomorrow morning.

Sentence Segmentation

County sheriffs chased Alex Jones of San Jose all the way to Gilroy, where he lived. Jones had run from a crime scene near his employer. The sheriff's department promised more details by tomorrow morning.



County sheriffs chased Alex Jones of San Jose all the way to Gilroy, where he lived.

Jones had run from a crime scene near his employer.

The sheriff's department promised more details by tomorrow morning.

Yes, use obvious end-sentence markers like “.”

Sentence Segmentation

However, look at this example:

> County sheriffs chased Mr. Jones of San Jose, Ca. all the way to Gilroy, where he lived. Jones had run from a crime scene near his employer, Big Haulers, Inc. The sheriff's dept. promised to provide more details by tomorrow morning.

What's harder about this one?

Sentence Segmentation

County sheriffs chased Mr. Jones of San Jose, Ca. all the way to Gilroy, where he lived. Jones had run from a crime scene near his employer, Big Haulers, Inc. The sheriff's dept. promised to provide more details by tomorrow morning.

County sheriffs chased Mr. Jones of San Jose, Ca. all the way to Gilroy, where he lived.

Jones had run from a crime scene near his employer, Big Haulers, Inc.

The sheriff's dept. promised to provide more details by tomorrow morning.

Fortunately, we have good *sentence tokenizers* that do this by examining punctuation, abbreviations, and capitalized words.

Word Tokenization

- Now that we have sentences broken out of a document, the next step is to grab individual words.
- This seems easy, right? Words are generally space-delimited, right?
- Not quite:

> The red fox didn't feel like jumping over the hyperactive poodle, so he just went around the "crazy" dog.

Word Tokenization

> The red fox didn't feel like jumping over the hyperactive poodle, so he just went around the "crazy" dog.

Using space separation only, the above tokenizes to:

> | The | red | fox | didn't | feel | like | jumping, | over | the | hyperactive | poodle, | so | he | just | went | around | the | "crazy" | dog. |

And what's wrong with that? Look closely. Did it yield nothing but *words*?

Word Tokenization

> The red fox didn't feel like jumping over the hyperactive poodle, so he just went around the "crazy" dog.

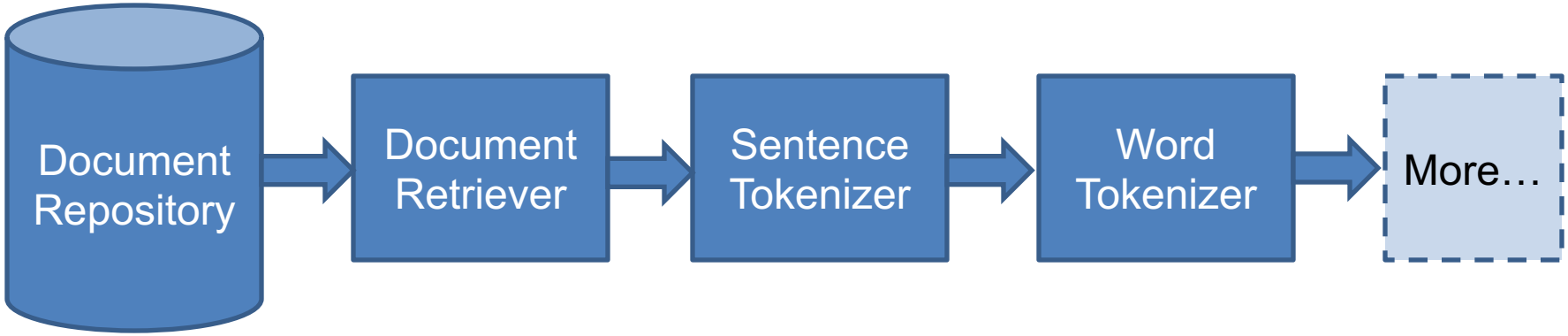
A real word tokenizer will give us something like this:

> | The | red | fox | did | "n't" | feel | like | jumping | over | the | hyperactive | poodle | so | he | just | went | around | the | crazy | dog |

- Notice the stripping of punctuation and quote marks, and the breaking out of the contraction.
- This makes it ready for further analysis.

Sequence of Tokenization

- Also notice the word tokenizer expects to be given (inputted) a sentence at a time.
- So we will generally run things in a certain sequence.



Typical NLP project, preprocessing phase

DataScience@SMU

Text Normalization

Natural Language Processing

Text Normalization Tasks

To normalize a word-tokenized text source, we often address the following:

- Contractions, expansion of
- Stop words, removal of
  - Understanding the difference between content words, function words, and stop words
- Misspellings, resolving of
- Stemming, if called for

Handling Contractions

Contractions are increasing in their frequency and variety in written text.

It's no longer just

"isn't"
"can't"

but also things like

"couldn't've"
"you'll've"

(This is especially true if you're dealing with UGC of course.)

So we have functions that utilize lexicons of contractions, so that we can expand them to full words.

Managing Stop Words

Stop words are words we want to discard because they are not informative to our application. We must understand the difference among:

- Content words
- Function words
- Stop words

When in the Course of human events it becomes necessary for one people to dissolve the political bands which have connected them with another and to assume among the powers of the earth, the separate and equal station to which the Laws of Nature and of Nature's God entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the separation.

Managing Stop Words

Stop words are words we want to discard because they are not informative to our application. We must understand the difference among:

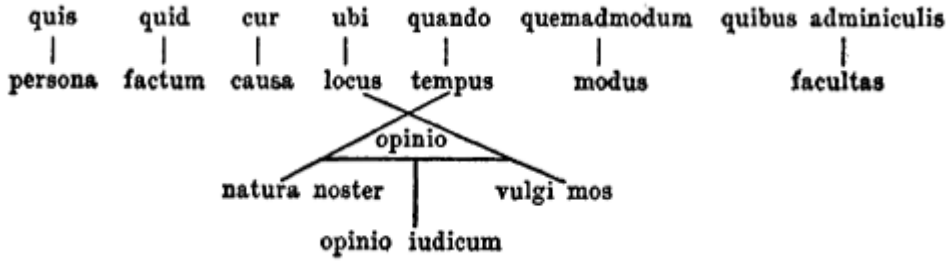
- Content words
- Function words
- Stop words

…Course …human events …necessary …one people …dissolve …political bands …connected …another …assume …powers …earth, …separate …equal station …Laws …Nature …Nature's God entitle …decent respect …opinions …mankind requires …declare …causes …impel …separation.

Content vs. Function Words

Function words…

- are a *closed class*, meaning they are a fixed list to which no new words are added.
- lack content, in that they don't answer the "6 Ws" of *who, what, why, when, where*, and *how* (though they may set up the structure for an answer).
- Examples are *is, am, are, was, were, he, she, you, we, they, if, then, therefore, possibly.*



A 4th century diagram mapping the "7 Qs" of Hermagoras to the "circumstances" of Cicero

Content vs. Function Words

Content words…

- are an *open class*, meaning they are an unfinished list to which new words are readily added.
- bear "content" in that they answer the "6 Ws" of *who, what, why, when, where*, and *how*.
- Examples are *dog, cat, angrily, run, clever, democracy, green, God, wisdom.*
- So the two sets of function and content words do not intersect, and function words are usually outnumbered by content words.

Function Words vs. Stop Words

- If you think the list of function words looks like a list of stop words, you're right.
- Our baseline list of stop words is usually just the list of all the function words in our language.

Function Words vs. Stop Words

- However, a stop word list is *application specific*, which means we usually add some content words to it that we want to discard.
- An example is that if we want to gather only personal names from text, we may add personal titles to the stop word list: *Mr., Mrs., Ms., Dr., Prof., Fr.*, etc.

Handling Misspellings

We wouldn’t want to miscount occurrences of a word because of misspellings. Consider these commonly misspelled words:

absence	committee	excellent	interrupt	pneumonia	similar
accidentally	conscience	excitement	judgment	possess	sincerely
accommodate	convenience	experience	leisure	practically	straight
achieve	courageous	familiar	library	preferred	strengthen
acquaintance	criticize	fascinate	lightning	privilege	studying
against	decision	favorite	lonely	probably	summarize
a lot	definitely	February	lying	raspberry	surprise
already	desperately	finally	millionaire	realize	thorough
argument	dictionary	foreign	mischievous	receive	tomorrow
attendance	different	friend	necessary	recommend	truly
because	disease	government	niece	remember	until
beginning	doesn't	grammar	noticeable	restaurant	vacuum
believe	embarrass	guarantee	occasion	rhythm	vegetable
business	enough	height	occurred	ridiculous	Wednesday
calendar	environment	immediately	opposite	schedule	weight
cemetery	especially	independent	particular	scissors	weird
chief	exaggerate	instead	physical	separate	yacht

Handling Misspellings

- Two fundamental approaches:
  - Edit-distance method
  - Fuzzy string compare
- Special short-circuits: some simple corrections can be made before invoking our main spell corrector
  - Repeated characters, e.g., “mostttly” → “mostly”
  - Can you think of other short-circuits?

Edit Distance Method

An edit is:

- Adding one character
- Deleting one character
- Replacing one character

Our basic algorithm returns the word with the lowest edit distance, up to a maximum distance (typically three).

Why a recommended max-edit-distance of about three?

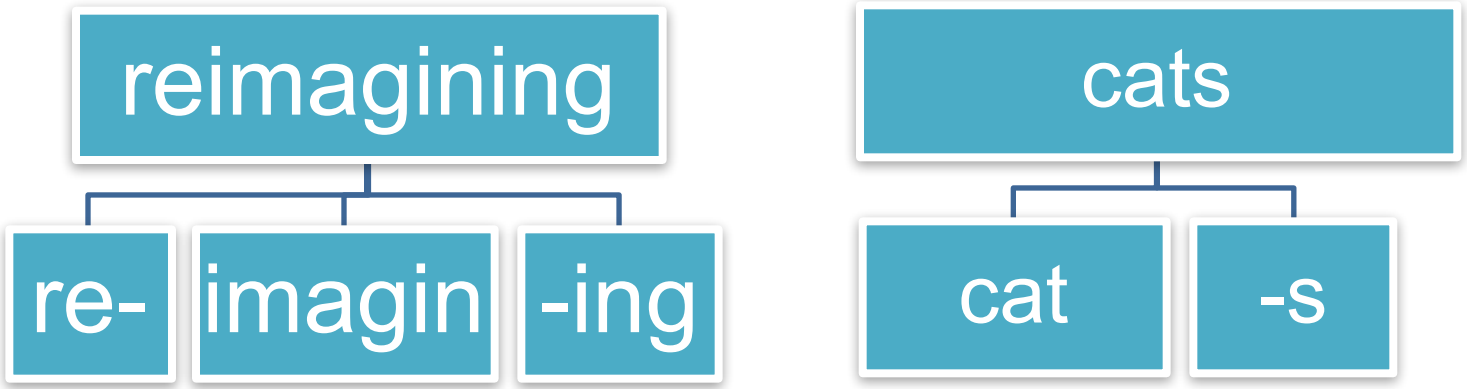
- The bigger we make this number, the slower our process (more computationally intensive, exponentially increasing search space).
- Also, it's less likely to give a meaningful, helpful result.

Fuzzy String Compare

- Characters-in-common as a percentage of total characters.
- Characters must be in like-order.
- We do this by finding the longest common substring, then separating the leftover (nonmatching) substrings to the left and right.
- Iterate to the left, then the right until no more characters remain to compare.

Stemming

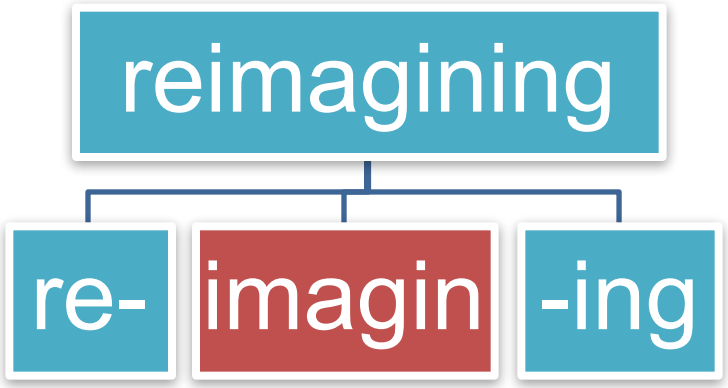
- Stemming is a task wherein NLP utilizes morphology. So, what's morphology?
- Morphology studies how words can be made out of smaller parts that actually have meaning (or have an impact on meaning).
- The smaller parts are called *morphemes*.



Stemming

Morphemes are of two main types:

- Stems
  - Supplies the main meaning
  - Can *sometimes* be a word in itself (e.g., “jump” in “jumping”)
- Affixes
  - Adds or alters meaning
  - Cannot usually be a word by itself



Stemming

A stemmer is tool that given a word, returns its stem:

“running” → “run”

“derivation” → “deriv”

“phenomenology” → “phenomenon”

“antidisestablishmentarianism” → “establish”

The Scope and Limits of Stemmers

Many stemmers are imperfect, but we can still use them to get a "fair" count of essentially the same word in a document.

For some applications, it doesn't matter that we are given "arriv" instead of "arrive" for all the variant forms ("arriving," "arrived," "arrival")—it still reduces them to one stem, so that they can be counted together.

Blah blah blah blah blah blah blah arrive blah blah blah blah blah blah blah arrived blah blah blah blah blah blah blah blah blah arriving blah blah blah blah blah and arrival of blah blah to blah blah when it arrives from blah blah blah blah

DataScience@SMU

Low-Level Document Feature Extraction

Natural Language Processing

Document Feature Extraction

Some low-level* feature extractions:

- Primary features
  - Require us only to examine the document itself
  - Examples:
    - Word frequencies
    - Collocations (*n*-grams)
- Secondary features
  - Require us to compare features of the document to those of other documents
  - Examples:
    - Differential frequency analysis (TF/IDF)
    - Relative lexical diversity
    - Reading level

*not requiring syntactic or semantic analysis

Terminology Extraction

Frequency-based term extraction is the simplest method of terminology extraction.

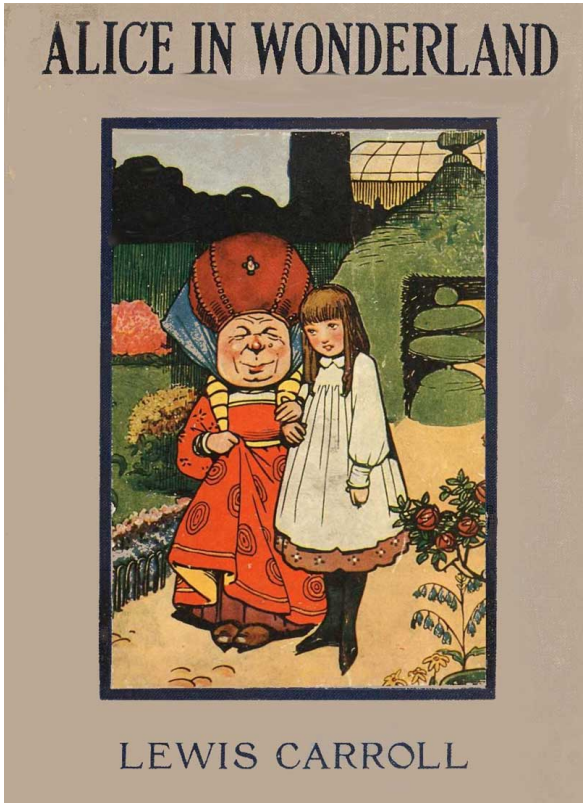
- The most common words in a document are taken to represent it.

- Initially, for *Alice in Wonderland*, it gives us this.

'the'
'and'
'to'
'a'
'of'
'I'
'it'
'she'
'said'
'in'
'Alice'
'you'
'was'
'that'
'as'
'her'
'with'
'at'
'on'
'all'

Terminology Extraction

Can be made smarter by discarding of stop words



'the'
'and'
'to'
'a'
'of'
'I'
'it'
'she'
'said'
'in'
'Alice'
'you'
'was'
'that'
'as'
'her'
'with'
'at'
'on'
'all'

Run-Ons, a.k.a. Collocations

- Extracted collocations, called *n*-grams (e.g., unigrams, bigrams, trigrams) are multiword phrases found in a text.
- The most common *n*-grams can be informative but may contain superfluous words.
- Notice the most common bigrams in *Alice in Wonderland*—what's helpful and what's not?

"said alice"
"mock turtle"
"march hare"
"said king"
"thought alice"
"said hatter"
"white rabbit"
"said mock"
"said gryphon"
"said caterpillar"

Differential Frequency Analysis

- To improve on the extraction of terms, we can use differential frequency analysis—noticing phrases that are more common in a target text than in most other texts.
- The most common technique is called "TFIDF" for "term frequency–inverse document frequency."

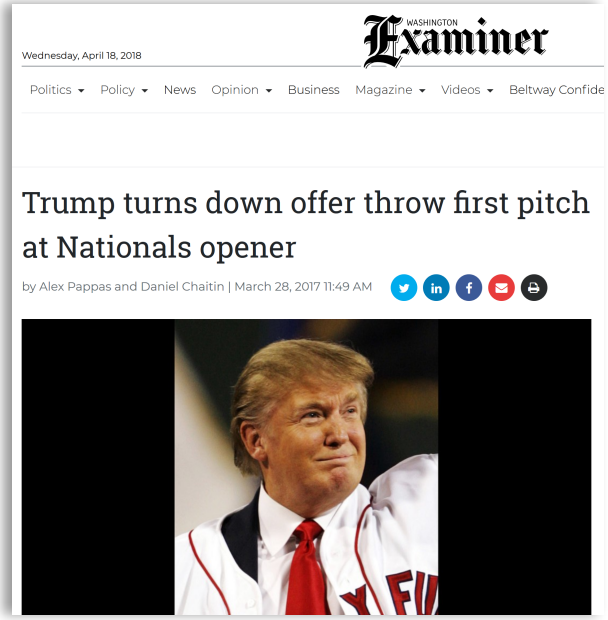
Freq-Rank of "turtle" (after discarding stop words) in:

Alice in Wonderland	*100 Classic Children's Books* (corpus)
11	*3,403*

Differential Frequency Analysis

- One virtue of this method is that it can show us what stands out *relative to a corpus.*
- Consider essentially the same story within two different corpora: *Sporting News* and *Washington Examiner.*

VS



TF-IDF: Term Frequency

- *Term frequency* measures how often a term occurs in a document—in raw form, it is a simply a word count divided by the total number of words in the document.
- *Document frequency* measures how common the term is within a domain represented by a corpus of documents (C*)*.
- *Inverse document frequency* is computed by dividing the total number of documents in the corpus by the number of documents containing our target term, and applying a log scale.

$$idf(t) = 1 + \log \frac{C}{1 + df(t)}$$

To avoid division by zero, we pretend every term occurs in at least one document of the corpus, e.g., we insert the “1+” in the denominator.

TF-IDF

Finally we can put it together for a TF-IDF score of any given term in any target document:

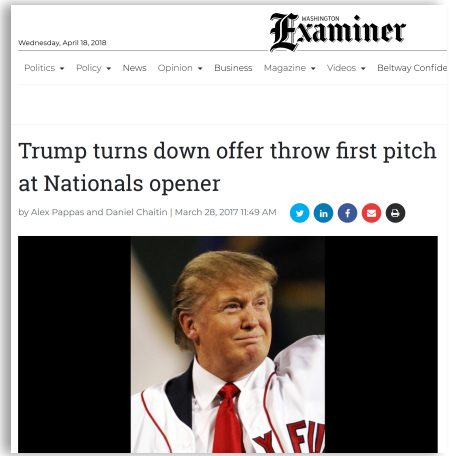
$$tfidf = tf \times idf$$

We'll automatically get a higher tfidf weight when there is a higher term frequency with a lower document frequency.

Differential Frequency Analysis

- TFIDF-based keyword extraction could look like this:

vs.



Trump
Commander in Chief
Taft
Nationals
politicians
White House
race
ethnicity

Washington Nationals
Miami Marlins
MLB
Taft
pitch
Trump
1910
White House

Secondary Features

- Secondary features require us to compare the primary features of a document to either
  - an external knowledge base, or
  - features of other documents
  - in order to derive a secondary feature of the original document.
- Examples:
  - Comparative lexical diversity
    - Breadth of vocabulary relative to document length
    - Compare our document's ratio to that of other documents
    - Application to writing help, as a cue for intervention/prompting
  - Readability level
    - Examine things such as sentence length, average number of syllables, words outside of a baseline list
    - Applications are numerous!

Dale-Chall Readability Formula

The New Dale-Chall Readability Formula

Compute the following equation:

Raw Score = 0.1579 * (PDW) + 0.0496 * ASL

Raw Score = Reading Grade of a reader who can comprehend your text at 3rd grade or below.

PDW = Percentage of Difficult Words

ASL = Average Sentence Length in words

If (PDW) is greater than 5%, then:

Adjusted Score = Raw Score + 3.6365, **otherwise** Adjusted Score = Raw Score

Adjusted Score = Reading Grade of a reader who can comprehend your text at 4th grade or above.

Step 5: Use the following table to get the Adjusted Grade Level:

ADJUSTED SCORE	GRADE LEVEL
4.9 and Below	Grade 4 and Below
5.0 to 5.9	Grades 5 - 6
6.0 to 6.9	Grades 7 - 8
7.0 to 7.9	Grades 9 - 10
8.0 to 8.9	Grades 11 - 12
9.0 to 9.9	Grades 13 - 15 (College)
10 and Above	Grades 16 and Above (College Graduate)

- "New" means it was updated in 1995, from the original 1948 formula.
- Key to the formula is a list of words familiar to 80% of 4th graders in 1995. "PDW" just means the % of words *not* on this list.
- It's one of the better readability formulas, but how could it be improved?

DataScience@SMU