BUS 243

Lecture 3: Text Representation

PROCESS

- Tokenization is a particular kind of document segmentation
  - Break up text into smaller chunks
  - Document → paragraphs → sentences → phrases → tokens
- Tokenization is the first step in an NLP pipeline
  - turns an unstructured string (text document) into a numerical data structure
- Let’s see the code

TERMINOLOGIES…

- "Type," "word," and "token" are terms used to describe different units of text
- Type: a unique distinct word or lexeme in a language, independent of inflections
  - *The cat chased the cat*
  - Two types: "cat" and "chased"
- Word: a grammatical unit that represents a single lexical unit of meaning
  - *I like to read books*
  - Five words
- A token is an instance or occurrence of a word
  - *I like to read books.*
  - 6 tokens including a punctuation
  - Use token and word interchangeably here

ONE-HOT VECTOR?

- What's the downside?
  - Storing all those zeros, and trying to remember the order of the words in all your documents, doesn't make much sense
- You'd like to compress your document down to a single vector rather than a big table
  - Trade-off: need to give up something
- We will revisit this representation later (sentence-base analysis)
  - Essential input for CNN (Convolution Neural net)
- Let's go back to the code

BAG OF WORDS

- A common approach is to use a column vector of word counts
  - $\boldsymbol{x} = [0,1,1,0,13, \ldots]^T$, where x_j is the count of word j
  - The length of $\boldsymbol{x}$ is the set of possible words in the vocabulary
- $\boldsymbol{x}$ is a vector, but it is often called a **bag of words**
  - Includes only information about the count of each word
  - NOT the order in which the words appear
  - Ignore grammar, sentence boundaries, paragraphs

TOKENIZATION

- The first subtask for constructing a BOW vector is **tokenization**
  - A sequence of characters → a sequence of **word tokens**
- Note whitespace-based tokenization is not ideal
- Tokenization is typically performed using regular expressions, with modules designed to handle each cases
  - Go back to the code example

TOKEN IMPROVEMENT

- See the text for the regular expressions
- See a number of tokenizers in the code examples
- Social media researchers have found that emoticons and other forms of orthographic variation pose new challenges for tokenization, leading to the development of special purpose tokenizers to handle them
  - O'Connor, B., M. Krieger, and D. Ahn (2010). Tweetmotif: Exploratory search and topic summarization for twitter. In Proceedings of the International Conference on Web and Social Media (ICWSM), pp. 384–385.

TOKENIZATION IS HARD?

- Tokenization is a language-specific problem
  - Each language poses unique challenges
    - Chinese does not include spaces between words, nor any other consistent orthographic markers of word boundaries
    - German does not include whitespace in compound nouns
- Social media raises similar problems for English and other languages
  - #TrueLoveInFourWords
  - Decomposition analysis (Brun and Roux, 2014)

EXTENDING YOUR VOCABULARY WITH N-GRAMS

- Now we consider a sequence of words
  - Ice cream
  - Boston Red Sox
- N-gram is simply a sequence of n words
  - N-gram could denote characters, but focus on words now
- We have tokenized sentences using 1-gram only thus far
- Using 2-gram or 3-gram words means adding more tokens in the vocabulary
  - Not difficult to add (see the codes)
  - N-gram tokens are pretty rare → need some ways to handle them properly

WHAT IF ONLY USE 1-GRAM TOKENS?

- What is the problem of rare 2-grams when we add them in the vocab?
  - Again, they are so rare. Why is this a problem?
- If use 1-gram tokens only, the stop words are usually counted the most
  - The, a, an, …
- If they are removed:
  - Mark **reported** to the **CEO**
  - Suzanne **reported** as the **CEO** to the board
    - Lack of information about the professional hierarchy
- If not, the length of vocabulary would be the problem
- Let's dig this issue a little bit deeper

TEXT NORMALIZATION

- After splitting the text into tokens, the next question:
  - Which tokens are really distinct
- Complete elimination of case distinction will result in a smaller vocab
  - Necessary to distinguish *great*, *Great* and *GREAT*?
  - How about *apple* and *Apple*?
- Text normalization refers to string transformation that remove distinctions that are irrelevant to downstream applications
  - Also include standardization of numbers (1,000 → 1000) or dates
  - Social media (e.g. cooooooool)

INFLECTIONS MATTER?

- A more extreme form of normalization is to eliminate inflectional affixes (e.g. -ed and –s suffixes)
  - Whale, whales, whaling all refer to the same underlying concept
- A stemmer is a program for eliminating affixes
  - Apply a series of regular expression substitutions
  - Character-based stemming algorithms are necessarily approximate

Original	**The**	**Williams**	**sisters**	**are**	**Leaving**	**This**	**Tennis**	**centre**
Porter stemmer	the	william	sister	are	leav	thi	tenni	centr
Lancaster stemmer	the	william	sist	ar	leav	thi	ten	cent
WordNet lemmatizer	The	Williams	sister	are	leaving	this	tennis	centre

INFLECTIONS MATTER?

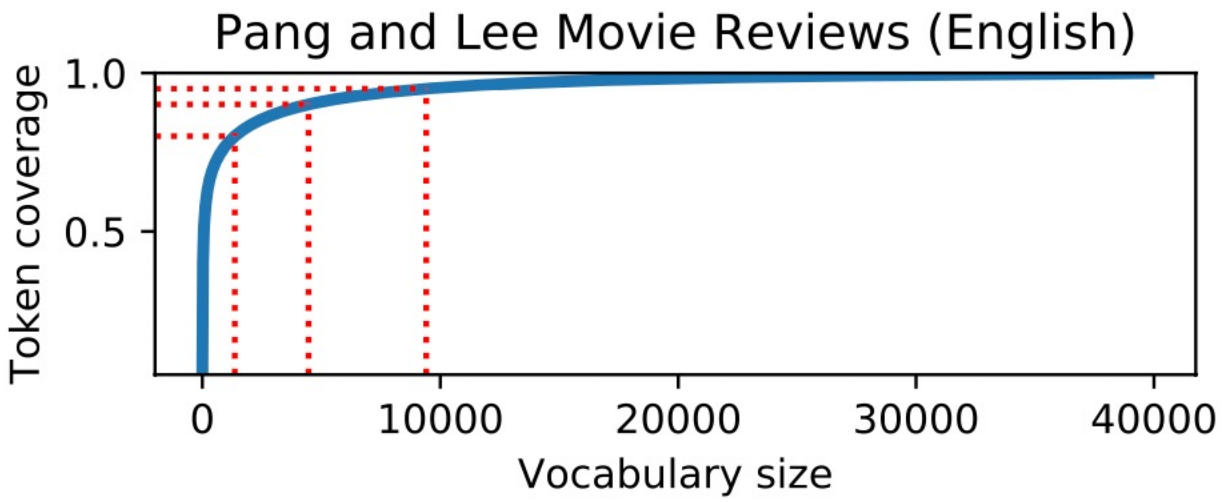
- **Lemmatizers** are systems that identify the underlying lemma of a given wordform
  - Geese → Goose
- Generalization would matter
  - Even inaccurate stemming can improve bag-of-words classification
    - merging related strings and thereby reducing the vocabulary size
  - However, need to avoid the over-generalization errors
- Both stemming and lemmatization are language-specific
  - English stemmer or lemmatizer is of little use on a text in another language

NORMALIZATION IS KIND OF SMOOTHING

- The value of normalization depends on the data and the task
  - Normalization reduces the size of the feature space
  - Can help in generalization
- There is always the risk of merging away meaningful distinctions
- In supervised machine learning, regularization and smoothing can play a similar role to normalization
  - Mitigate overfitting to rare (language-specific) features
- In unsupervised learning, such as topic modeling, normalization is even more critical

HOW MANY WORDS?

- Limiting the size of the feature vector reduces the memory and increases the speed of prediction
- Normalization can help to play this role, but a more direct approach is simply to limit the vocabulary to the ***N*** most frequent words in the dataset



STOPWORDS

- Another way to reduce the size is to eliminate stopwords (the, to,…)
  - Typically done by creating a stoplist (nltk stopwords) and ignoring all terms that match the list
- However, seemingly inconsequential words can offer surprising insights about the author or nature of the text (Biber 1991)
- High-frequency words are unlikely to cause overfitting in discriminative classifiers
- As with normalization, stopword filtering is more important for unsupervised problems, such as term-based document retrieval