Introduction to Natural Language Processing

BUS 243 F: Spring 2023

Yeabin Moon

Lecture 2

Text Classification

- We begin with the problem of text classification
  - Given a text document, assign it a discrete label $y \in Y$, where Y is the set of possible labels
- Text classification has many applications
  - Spam filtering, open-ended survey analysis, health records analysis, etc.
- It involves a number of design decisions
  - The decision is sometimes clear from the mathematics
  - Other decisions are more subtle, arising only in the low-level “plumbing” code
    - How to process raw data

Goal of the lecture

- Understand a numerical representation of text
  - One-hot vectors
  - Bag-of-words representation

Some keywords

- 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 suitable for machine learning
- The distribution of tokens can be used directly
  - Counts imply the meanings
- They also used in a ml pipeline as features

Terminology alert: one-hot vectors

- One-hot vectors representation
  - Definition
  - Characteristics
  - What tokenizer used in the example?
- This representation of a sentence in one-hot word vectors retains all the detail, grammar, and order of the original sentence.

Need another approach

- 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
  - Essential input for CNN (Convolution Neural net)

Bag of words

- A common approach is to use a column vector of word counts
  - $\boldsymbol{x} = [0,1,1,0,13, \dots]^T$, 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

BOW: Everything but words

- The BOW model is surprisingly effective for text classification
- If you see the word *whale* in a document, is it fiction or nonfiction?
- For many labeling problems, individual words can be strong predictors
- Let’s see the code example

What is a word?

- The BOW representation presupposes that extracting a vector of word counts from text is unambiguous
- However, text documents are generally represented as a sequences of characters, and the conversion to bag of words presupposes a definition of the “words” that are to be counted

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. cooooooooool)

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?

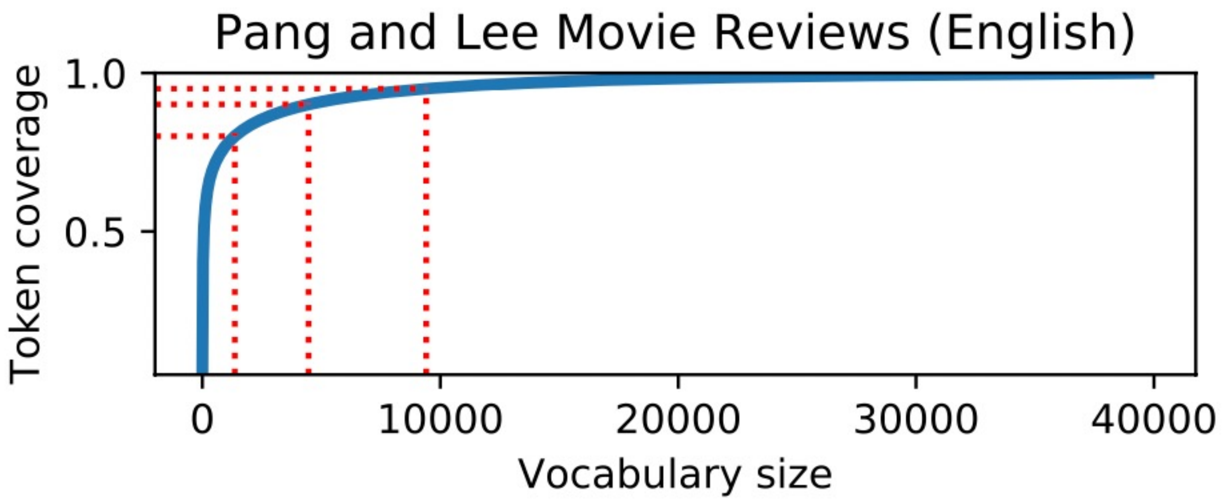
- **Lemmatizers** are systems that identify the underlying lemma of a given wordform
  - Geese → Goose
- Generalization would matter
  - Even inaccurate stemming can improve bag-of-words classification
    - merging related strings and thereby reducing the vocabulary size
  - However, need to avoid the 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

Count or binary?

- Consider whether we want to our vector to include the **count** of each word or its **presence**
- Pang et al. (2002) shows binary indicators performs better in some case
  - Words tend to appear in clumps: if a word has appeared once in a document, it is likely to appear again
  - These subsequent appearances can be attributed to this tendency towards repetition
  - Counts provide little additional information about the class label document

Sentiment (opinion) analysis

- A popular application of text classification is to automatically determine the **sentiment** or **opinion polarity** of documents
  - product reviews and social media posts
- Numerous application both in academia and practice
  - Macro-finance indicators from news or policy statements
  - Mood change by the weather reports
- Assume reliable labels can be obtained
- In simple case, it is a two or three-class problem
  - Positive, negative, or neutral

Popular annotations

- Tweets containing happy emoticons can be marked as positive, sad emoticons as negative (Read, 2005; Pak and Paroubek, 2010).
- Reviews with four or more stars can be marked as positive, three or fewer stars as negative (Pang et al., 2002).
- Statements from politicians who are voting for a given bill are marked as positive (towards that bill); statements from politicians voting against the bill are marked as negative (Thomas et al., 2006)

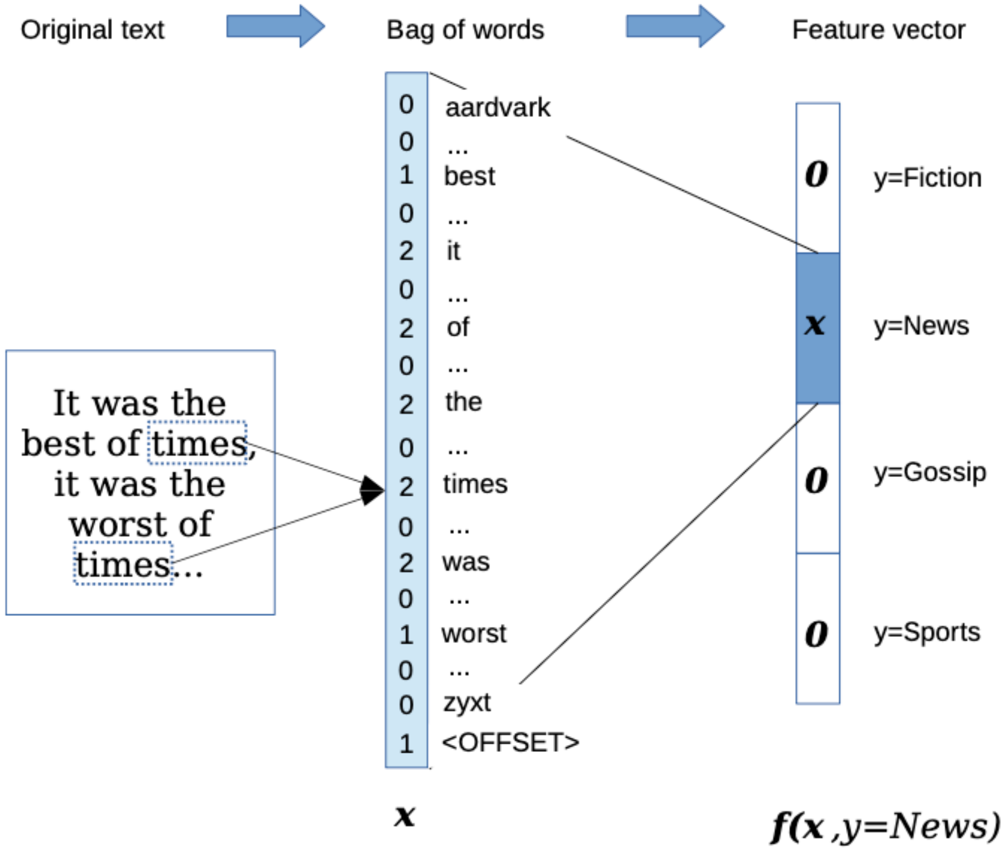
VADER

- VADER, Valence Aware Dictionary for sEntiment Reasoning
  - Never has beat Darth Vader
  - Easy to use
  - Limitations are so obvious
  - Its intuition would help for understanding distributed semantics (lecture 6)
- Let’s review the code

Sentiment analysis with the bag-of-words

- A Naive Bayes model tries to find keywords in a set of documents that are predictive of your target (output) variable
- When your target variable is the sentiment you are trying to predict, the model will find words that predict that sentiment
- The nice thing about a Naive Bayes model is that the internal coefficients will map words or tokens to scores just like VADER does

The bag-of-words for text classification task



Naïve Bayes

- The joint probability of a BOW $\boldsymbol{x}$ and the label y is written $p(\boldsymbol{x}, y)$
- Suppose we have N instances, which we assume IID, then
  - $\Pr\left(\boldsymbol{x}^{(1:N)}, y^{(1:N)}\right) = \Pi_{i=1}^{N} \Pr_{X,Y}(\boldsymbol{x}^{(i)}, y^{(i)})$
- What does this have to do with classification?
- One approach to classification is to set the weights θ to maximize the joint probability of a training set of labeled documents
  - Known as maximum likelihood estimation
  - $\hat{\theta} = argmax_{\theta} \Pi_{i=1}^{N} Pr(\boldsymbol{x}^{(i)}, y^{(i)}; \theta)$

Assumptions in words

- The instances are mutually independent
  - Neither the label nor the text of document i affects the label or text of document j
- The instances are identically distributed
  - The distributions over the label $y^{(i)}$ and the text $\boldsymbol{x}^{(i)}$ are the same for all in all instances i
  - That is, every document has the same distribution over labels, and that each document's distribution over words depends only on the label, and not on anything else about the document
- The documents don't affect each other
- Now, let's see the codes

Discussion

- The bag-of-words model is a good fit for sentiment analysis at the document level
  - If the document is long enough, we would expect the words associated with its true sentiment to overwhelm the others
- But it is less effective for short documents, such as single-sentence reviews or social media posts
  - Linguistic issues like negation are inevitable

Guess sentiment

- That’s not bad for the first day
- This is not the worst thing that can happen
- It would be nice if you acted like you understood
- This film should be brilliant. The actors are first grade. It should like a great plot, however, the film is a failure. (Pang et al., 2002)

Discussion

- A minimal solution is to move from a bag-of-words model to a bag-of-bigrams model, where each base feature is a pair of adjacent words
  - (that’s, not), (not, bad), (bad, for),…
- Bigrams can handle relatively straightforward cases, such as when an adjective is immediately negated
  - But this approach will not scale to more complex examples
- Smoothing would be another option
  - Naïve Bayes use Laplace smoothing
  - What’s the reasoning behind smoothing in ML?
  - Let’s see the code