Introducing to Natural Language Processing and Text Classification

Goal: discuss how to build a spam filter.

documents come in and we decide \{spam, ham\}

We will see how to train a classifier using prelabeled documents (in this case, emails w/ 'spam' or 'ham' labels)

To do this requires knowing:

1. How to preprocess text
2. How to turn text data into features (the $X$ in the equation $Y = f(X) + \epsilon$)
3. Probability theory to build a classifier.
   - Bayes theorem
   - Laplace smoothing, log-probabilities (implementation details)

Processing text

We've done this in the past (homeworks) but it is a key step to NLP (Natural Language Processing).

Terminology:

Token - words, punctuation, numbers, etc.
Sentence - ordered sequence of tokens

Tokenization - process of segmenting a sentence into tokens. Whitespace makes tokenizing English easy, easy. But other languages, such as Chinese, are hard to tokenize.

Corpus - A body of text, usually w/ a large number of sentences.
Documents - Many corpora are broken down into smaller bodies of text called documents, for example, each email is a document in an email corpus.

Parts-of-Speech Tag (POS) - Words may be nouns, verbs, adjectives, etc. POS tags are symbols representing those categories:

- **NN** - Noun
- **VB** - Verb
- **JJ** - Adjective
- **AT** - Article
- etc...

Stop words - common, low-meaning words "the", "that", etc. which are often, but not always removed from a text.

POS tagging - labelling the tokens in a sentence

The ball is red.

AT NN VB JJ

Stemming - breaking words down to root morphemes by, among other things, removing suffixes.

cats, catlike, catly ⇒ cat.

argue, argues, arguing, argued, ⇒ argue (not a word)

Lemmatization - grouping together the different inflected forms of a word:

- walk, walks, walked, walking ⇒ walk (same as stem)
- better ⇒ good (lemma not matched by stemming)

"meeting" may be a noun or a verb (lemma: meet). Context required.

Stemming and lemmatization are common text processing steps. ⇒ Normalize the words.
Turning text into features:

The simplest feature transform for a tokenized text is the Bag-of-Words model.

→ throw out any structure/order to the text and assume all the data is with the counts: i.e. the number of times each unique word occurred.

→ each document becomes a count vector

\[ y(\text{"foo")} = 7 \rightarrow \text{"foo" appeared in document d a total of 7 times}, \]

→ dimensionality of v's = # unique words in the corpus.

(Notice the effects stemming/lemmatization may have.)

→ unigrams \(\rightarrow\) n-grams.

Text classification \(\rightarrow\) spam filtering

Given a document d, what is the probability it is or is not spam?

We want \(Pr(\text{spam}|d)\). We can generalize to more than two categories/classes.

\[ Pr(c \mid d) \quad \text{where, for now, \(c \in \{\text{spam, ham}\}\).} \]

If we can compute this probability for each \(c\), we can decide which class the document belongs to.

\[ C_{\text{MAP}} = \arg \max_c P(c | d) \quad (\text{MAP = maximum a posteriori, most likely}) \]

But, how to compute \(Pr(c | d)\)?