Text Classification (Part IV)

[DAT640] Information Retrieval and Text Mining

Krisztian Balog
University of Stavanger

September 3, 2019

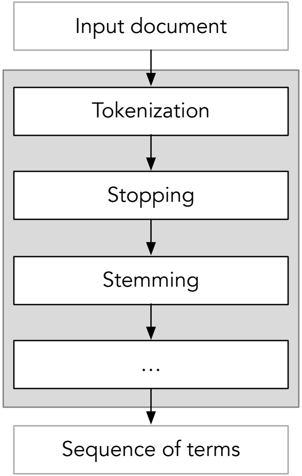
Recap

- Text classification
  - Problem, binary and multiclass variants
  - Evaluation measures
  - Training text classifiers using words (terms) as features
  - Term weighting (TFIDF)

Today

- Text preprocessing
- Non-term-based features for text classification
- Implementing a Naive Bayes classifier from scratch

Text preprocessing pipeline



Tokenization

- Parsing a string into individual words (tokens)
- Splitting is usually done along white spaces, punctuation marks, or other types of content delimiters (e.g., HTML markup)
- Sounds easy, but can be surprisingly complex, even for English
  - Even worse for many other languages

Discussion

Question

What could be the issues with tokenization along whitespace and punctuation marks?

Tokenization issues

- Apostrophes can be a part of a word, a part of a possessive, or just a mistake
  - *rosie o'donnell, can't, 80's, 1890's, men's straw hats, master's degree, ...*
- Capitalized words can have different meaning from lower case words
  - *Bush, Apple, ...*
- Special characters are an important part of tags, URLs, email addresses, etc.
  - *C++, C#, ...*
- Numbers can be important, including decimals
  - *nokia 3250, top 10 courses, united 93, quicktime 6.5 pro, 92.3 the beat, 288358, ...*
- Periods can occur in numbers, abbreviations, URLs, ends of sentences, and other situations
  - *I.B.M., Ph.D., www.uis.no, F.E.A.R., ...*

Stopword removal

- Function words that have little meaning apart from other words: *the*, *a*, *an*, *that*, *those*, ..
- These are considered **stopwords** and are removed
- A stopwords list can be constructed by taking the top-k (e.g., 50) most common words in a collection
  - May be customized for certain domains or applications

Example (minimal stopword list)

a	as	by	into	not	such	then	this	with
an	at	for	is	of	that	there	to	
and	be	it	it	on	the	these	was	
are	but	in	no	or	their	they	will	

Discussion

Question

What about a text like "to be or not to be"?

Stemming

- Reduce the different forms of a word that occur to a common stem
  - Inflectional (plurals, tenses)
  - Derivational (making verbs nouns etc.)
- In most cases, these have the same or very similar meanings
- Two basic types of stemmers
  - Algorithmic
  - Dictionary-based

NLTK

- Natural Language Toolkit (NLTK) – `https://www.nltk.org/`
- Leading Python library for natural language processing
- Working with text corpora, tokenization, analyzing linguistic structure, etc.

Exercise #1

- Create a term vector representation of an email message (i.e., any data file from Assignment 1)
  1. Use `sklearn`'s `CountVectorizer`
  2. Use NLTK's Porter stemmer
- Consider text both in the subject and body fields
- Compare the two vocabularies created from that single email
- Compare the size of the vocabularies on a larger set of emails
- Code skeleton on GitHub: `exercises/lecture_05/exercise_1.ipynb` (make a local copy)

Discussion

Question

Can you think of non-term-based features for SPAM detection?

Non-term-based features for SPAM detection

- Presence of an attachment
- Presence of images
- Presence of JavaScript code
- Whether reply-to is specified/different from sender
- Time when the email was sent (day of week, hour, minute)
- Number of URLs / unique URLs in the email
- Number of capitalized words in email subject
- ...

Naive Bayes classifier

Naive Bayes classifier

- Estimating the probability of document $\boldsymbol{x}$ belonging to class y

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

- $P(\boldsymbol{x}|y)$ is the class-conditional probability
- $P(y)$ is the prior probability
- $P(\boldsymbol{x})$ is the evidence (note: it's the same for all classes)

Naive Bayes classifier

- Estimating the class-conditional probability $P(y|\boldsymbol{x})$
  - $\boldsymbol{x}$ is a vector of term frequencies $\{x_1, \ldots, x_n\}$

$$P(\boldsymbol{x}|y) = P(x_1, \ldots, x_n|y)$$

- "Naive" assumption: features (terms) are independent:

$$P(\boldsymbol{x}|y) = \prod_{i=1}^{n} P(x_i|y)$$

- Putting our choices together, the probability that $\boldsymbol{x}$ belongs to class y is estimated using:

$$P(y|\boldsymbol{x}) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Naive Bayes classifier

- How to estimate $P(x_i|y)$?
- Maximum likelihood estimation: count the number of times a term occurs in a class divided by its total number of occurrences

$$P(x_i|y) = \frac{c_{i,y}}{c_i}$$

  - $c_{i,y}$ is the number of times term x_i appears in class y
  - c_i is the total number of times term x_i appears in the collection
- But what happens if $c_{i,y}$ is zero?!

Smoothing

- Ensure that $P(x_i|y)$ is never zero
- Simplest solution:[1] Laplace ("add one") smoothing

$$P(x_i|y) = \frac{c_{i,y} + 1}{c_i + m}$$

  - m is the number of classes

[1] More advanced smoothing methods will follow later for Language Modeling

Practical considerations

- In practice, probabilities are small, and multiplying them may result in numerical underflows
- Instead, we perform the computations in the log domain

$$\log P(y|\boldsymbol{x}) \propto \log P(y) + \sum_{i=1}^{n} \log P(x_i|y)$$

Exercise #2

- Implement a Naive Bayes text classifier
- Code skeleton on GitHub: `exercises/lecture_05/exercise_2.ipynb` (make a local copy)