Text Classification (Part III)

[DAT640] Information Retrieval and Text Mining

Krisztian Balog
University of Stavanger

August 27, 2019

Recap

- Implementing a text classification model using `scikit-learn`
  - GitHub: `code/text_classification.ipynb`
- Word counts used as features
- Document-term matrix is huge, but most of the values are zeros; stored as a sparse matrix

	t_1	t_2	t_3	$\dots$	t_m
d_1	1	0	2		0
d_2	0	1	0		2
d_3	0	0	1		0
$\dots$					
d_n	0	1	0		0

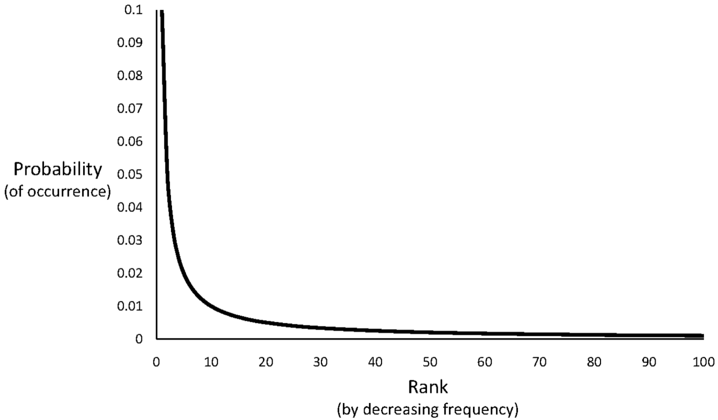
Discussion

Question

What are possible shortcomings of using raw term frequencies?

Zip's law

- Given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table
  - Word number n has a frequency proportional to $1/n$



English language

- Most frequent words
  - the (7%)
  - of (3.5%)
  - and (2.8%)
- Top 135 most frequent words account for half of the words used

Term weighting

- Intuition #1: terms that appear often in a document should get high weights
  - E.g., The more often a document contains the term "dog," the more likely that the document is "about" dogs
- Intuition #2: terms that appear in many documents should get low weights
  - E.g., stopwords, like "a," "the," "this," etc.
- How do we capture this mathematically?
  - Term frequency
  - Inverse document frequency

Term frequency (TF)

- We write $c_{t,d}$ for the raw count of a term in a document
- **Term frequency** $tf_{t,d}$ reflects the importance of a term (t) in a document (d)
- Variants
  - Binary: $tf_{t,d} \in \{0, 1\}$
  - Raw count: $tf_{t,d} = c_{t,d}$
  - **L1-normalized**: $tf_{t,d} = \frac{c_{t,d}}{|d|}$
    - where $|d|$ is the length of the document, i.e., the sum of all term counts in d: $|d| = \sum_{t \in d} c_{t,d}$
  - L2-normalized: $tf_{t,d} = \frac{c_{t,d}}{||d||}$
    - where $||d|| = \sqrt{\sum_{t \in d} (c_{t,d})^2}$
  - Log-normalized: $tf_{t,d} = 1 + \log c_{t,d}$
  - ...
- By default, when we refer to TF we will mean the L1-normalized version

Inverse document frequency (IDF)

- **Inverse document frequency** idf_t reflects the importance of a term (t) in a collection of documents
  - The more documents that a term occurs in, the less discriminating the term is between documents, consequently, the less "useful"

$$idf_t = \log \frac{N+1}{n_t}$$

  - where N is the total number of documents in the collection and n_t is the number of documents that contain t
  - Log is used to "dampen" the effect of IDF

Term weighting (TF-IDF)

- Combine TF and IDF weights by multiplying them:

$$\textit{tfidf}_{t,d} = tf_{t,d} \cdot idf_t$$

  - Term frequency weight measures importance in document
  - Inverse document frequency measures importance in collection

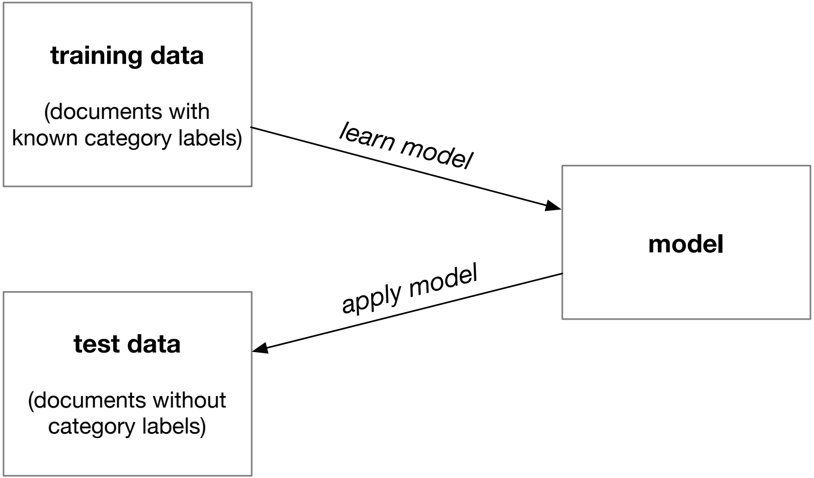
Exercise #1 (paper-based)

Create document-term matrix using TF-IDF weighting from a set of documents.

Code

- GitHub: `code/text_feature_extraction.ipynb`

Text classification



Text classification

- Formally: Given a training sample of documents $\boldsymbol{X}$ and corresponding labels y, $((\boldsymbol{X}, y) = \{(x_1, y_1), \dots (x_n, y_n)\})$, build a model f that can predict the class $y' = f(x)$ for an unseen document x
- Two popular classification models:
  - Naive Bayes
  - SVM

Exercise #2 (coding)

- Compare two machine learning models and different term weighting schemes
  - Naive Bayes and SVM
  - Raw term count, TF weighting, and TF-IDF weighting
- Complete the TODOs and fill out the results table
  GitHub: `exercises/lecture_04/exercise_2.ipynb` (make a local copy)

Assignment 1B