INTRODUCTION TO TEXT MINING
INF 582
Lab 3: Supervised document classification

Konstantinos Skianis, Antoine Tixier, Apostolos Papadopoulos
Michalis Vazirgiannis

January 27, 2017

1 Text categorization

Text categorization is one of the most active research areas in NLP. It has a variety of real-world applications such as sentiment analysis, opinion mining, email filtering, etc. Given the current data overflow, especially of textual type, the needs for efficient automated text classification solutions have become more pressing than ever.

The most common pipeline for text classification is the vector space representation followed by TF-IDF term weighting. With this approach, each document is viewed as a point in a sparse space where each dimension (or feature) is a unique term in the corpus. The set of unique terms $T = \{t_1 \dots t_n\}$ is called the vocabulary. If we assume there are m documents $\{d_1 \dots d_m\}$ in the collection and $n = |T|$ unique terms in the corpus, each document can be represented as a vector of n term weights (i.e., the weights are the coordinates of the document in the vocabulary space).

A classifier is then trained on the available documents (i.e., on a document-term matrix of dimension m by n) and subsequently used for classifying new ones. In this lab session, we are going to investigate different classifiers and how we can improve categorization performance by coming up with better term weights. We will compare two ways of computing these weights.

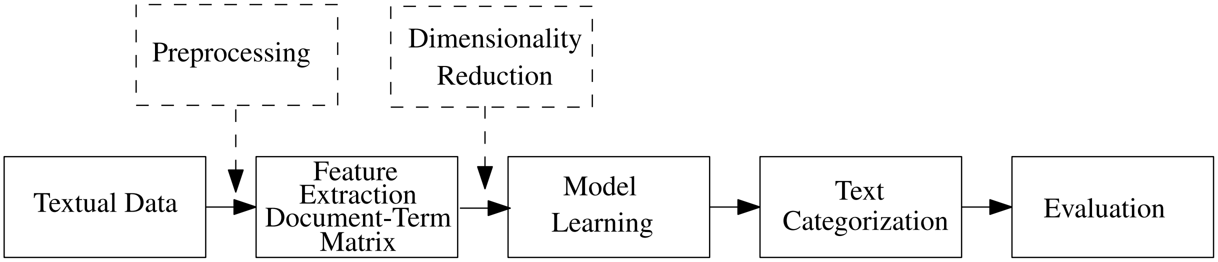

Figure 1: Basic pipeline of the Text Categorization task.

2 Feature extraction

2.1 Term Frequency-Inverse Document Frequency (TF-IDF)

A given document d loads on each dimension of the feature space (each term t) according to the following formula, known as the pivoted normalization weighting (Singhal et al. [1996]):

$$tf(t,d) = \#t \in d$$

$$idf(t,\mathcal{D}) = \log \frac{m+1}{df(t)} = \log \frac{m+1}{|\{d \in \mathcal{D} : t \in d\}|}$$

Then the tf-idf formula becomes:

$$tf\text{-}idf(t,d) = \frac{1 + \ln\big(1 + \ln(tf(t,d))\big)}{1 - b + b \times \frac{|d|}{avgdl}} \times idf(t,\mathcal{D})$$

Where $tf(t,d)$ is the number of times term t appear in document d, $|d|$ is the length of document d, $avgdl$ is the average document length across the corpus, $b = 0.2$, and $df(t)$ is the number of documents in which the term t appears.

The intuition behind this scoring function is that frequent words in a document are representative of that document as long as they are not also very frequent at the corpus level. Note that for all the terms that do not appear in d, the weights are null. Since a given document contains only a small fraction of the vocabulary, most of its coordinates in the vector space are null, leading to a very sparse representation, which is a well-known limitation of the vector space model that motivated (among other things) word embeddings, that we will address in the next lab session.

2.2 Term Weighting-Inverse Document Frequency (TW-IDF)

We propose to leverage the graph-of-words representation of a document (recall the previous lab session) to derive a new scoring function, TW-IDF (following Rousseau and Vazirgiannis [2013]), where node centrality metrics are used to replace the $tf(t,d)$ term at the numerator, and $b = 0.003$.

$$tw\text{-}idf(t,d) = \frac{tw(t,d)}{1 - b + b \times \frac{|d|}{avgdl}} \times idf(t,\mathcal{D})$$

Some basic centrality metrics that can intuitively be used as term weights are:

- *Normalized Degree.* The degree is a local metric equal to the number of incident edges of a node. The normalized degree can be computed as follows:

  $$\text{degree}(i) = \frac{|\mathcal{N}(i)|}{|V| - 1}.$$

  where $\mathcal{N}(i)$ is the number of connections of node i. Note that in its weighted version, the degree of a node is equal to the sum of the weights of its incident edges. Even though we won't be using it today, it is also possible to consider only the incoming edges of the node, or its outgoing edges, giving respectively the in-degree and out-degree centrality measures.

- *Closeness centrality.* It measures how close a node is to all the other nodes in the graph. It is formally defined as the inverse of the average shortest path distance from the node to any other node in the graph:

  $$\text{closeness}(i) = \frac{|V| - 1}{\sum_{j \in V} dist(i, j)}.$$

  Where $dist(i, j)$ is the shortest path distance between nodes i and j.

  Contrary to degree centrality, the closeness score is a global metric, in that it combines information from all the nodes of the graph. Here we compute the closeness centrality in the undirected graph.

3 Coding time

Each of the steps below is implemented either entirely or partially in the `document_classification.py` file that can be found under the `code` directory. The custom functions used can be found in the `library.py` file.

- We will work on a standard, publicly available dataset: WebKB. This data set corresponds to academic webpages belonging to four different categories: (1) project, (2) course, (3) faculty, and (4) students. It contains 2,803 documents for training and 1,396 for testing. The training and testing sets can be found under the `data` directory. They already have been preprocessed: stopwords and words less than 3 characters in length have been removed, and Porter' stemming has been applied, so that only the root of each term remains.

- **CODE:** Create a TF-IDF and TW-IDF representation for each document in the training set. For TW-IDF, build a graph-of-words for each document, with a window size of 4, (you can use the `terms_to_graph()` function found in `library.py`) and consider both the weighted and unweighted versions of degree and closeness.

- **CODE:** Train a basic Naive Bayes classifier (McCallum and Nigam [1998]).

  Try to use other classifiers like Multinomial Logistic Regression (Collins et al. [2002]) and linear kernel SVM (Joachims [1998]).
  Question: What about using more complex models?

- **CODE:** Use the classifiers previously trained to predict the labels of the documents in the test set. Note that the documents in the testing set should be represented in the space made of the unique terms in the training set *only* (**words in the testing set absent from the training set are disregarded**). Similarly, the average document length and the inverse document frequency used should be the ones computed on the training set.

- Compare performance for different features and one classifier. Which are the best features and why?

- Compare performance for the same features but different classifiers. Which is the best classifier and why?

- **CODE:** See the most important features of the classifier for each class, as well as the less important ones. Complete the `library.py` file by creating a function that outputs the 10 highest and lowest weights of a classifier for each class.

- Should we invest time on feature extraction or model selection?

References

Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. *Mach. Learn.*, 48(1-3):253–285, September 2002. ISSN 0885-6125. doi: 10.1023/A:1013912006537. URL http://dx.doi.org/10.1023/A:1013912006537.

Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK, 1998. Springer-Verlag. ISBN 3-540-64417-2. URL http://dl.acm.org/citation.cfm?id=645326.649721.

Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification, 1998.

François Rousseau and Michalis Vazirgiannis. Graph-of-word and tw-idf: New approach to ad hoc ir. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 59–68, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2263-8. doi: 10.1145/2505515.2505671. URL http://doi.acm.org/10.1145/2505515.2505671.

Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, pages 21–29, New York, NY, USA, 1996. ACM. ISBN 0-89791-792-8. doi: 10.1145/243199.243206. URL http://doi.acm.org/10.1145/243199.243206.