ADVANCED LEARNING FOR TEXT AND GRAPH DATA

Lab session 2: Supervised document classification

Lecture: Prof. Michalis Vazirgiannis
Lab: Antoine Tixier and Kostas Skianis

November 23, 2018

1 Introduction

Text categorization is one of the most active research areas in NLP. It has a variety of real-world applications such as sentiment analysis and opinion mining, email filtering, etc. It is also tightly connected to the field of information retrieval (i.e., search engines).

The most common pipeline for text classification is the vector space representation (a.k.a. "bag-of-words") followed by TF-IDF term weighting. With this approach, each document d_i from the collection $D = \{d_1 \dots d_m\}$ is viewed as an n-dimensional vector, where n is the number of unique terms in the processed collection. The set of unique terms $T = \{t_1 \dots t_n\}$ is called the vocabulary.

A classifier is then trained on the document-term matrix of dimension m by n, and is then used for classifying new documents. In this lab session, we will compare how using different weighting schemes and different classifiers impacts classification accuracy.

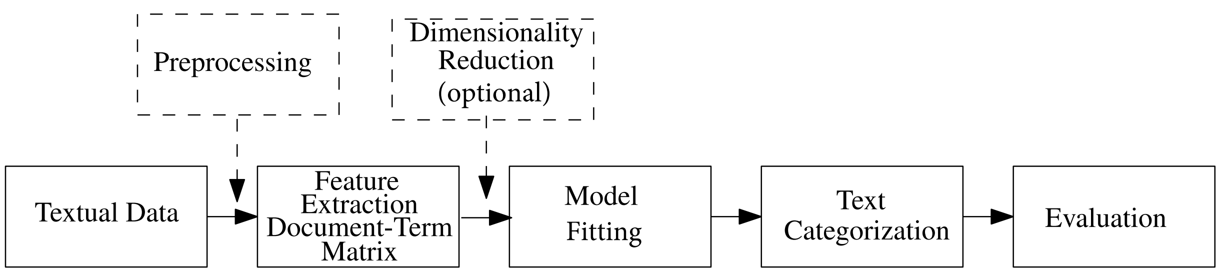

Figure 1: Basic pipeline of the Text Categorization task.

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

The coordinates of document d in the "bag-of-words" space (made of the $\{t_1 \dots t_n\}$ features) are typically computed as:

$$\text{weight}(t, d) = tf(t, d) \times idf(t, D)$$

Where $tf(t, d)$ is the number of times term t appears in document d, and $idf(t, D) = \log\left({}^{m+1}/_{df(t)}\right)$, with $df(t)$ the number of documents containing t and m the size of the collection D. 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 (i.e., the vector space is sparse).

Using the raw term frequency tends to favor long documents as they usually contain many repetitions of the same terms (which increases tf). Also, long documents contain more unique terms, which may artificially increase their similarity with other documents. In an attempt to fix the aforementioned issues (in the context of information retrieval), Singhal et al. [1996][1] proposed a new term frequency formula, known as the *pivoted normalization weighting*:

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

Where $|d|$ is the length of document d, $avgdl$ is the average document length across the corpus, and $b = 0.2$. The use of a concave function in the numerator repeatedly diminishes the gain of seeing an additional occurrence of a term in a document (the gain of increasing tf from 1 to 2 should be much larger than the gain of increasing tf from 1000 to 1001). In the denominator, $b \in [0,1]$ is a tuning parameter that controls how much document length should be taken into account. The smaller b, the less normalization by document length. Regardless of the value of b, this approach assumes that a document of average length is of appropriate size and does not need normalization.

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

We capitalize on the graph-of-words representation introduced during the previous lab session to derive a new scoring function, TW-IDF (Rousseau and Vazirgiannis [2013]), where node centrality metrics are used to replace the numerator of $tf_p(t,d)$, and $b = 0.003$.

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

Where G_d is the graph-of-words representation of document d. The absence of concave normalization in the numerator and the very low value of b are justified in the case of graphs-of-words with unweighted edges. Indeed, in that case, edge weights (and thus node metrics) do not increase linearly with the size of the document like raw term frequency. They only increase when new edges are added, which corresponds to new contexts of co-occurrence and depends much less on document size.

Some basic centrality metrics that can intuitively be used to provide the term weights $tw(t,G_d)$ are the:

- *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:

$$\text{degree}_G(v) = \frac{|\mathcal{N}(v)|}{|G| - 1}$$

[1] Amit Singhal was the head of Google's core search team from 2000 to 2016.

where $\mathcal{N}(v)$ is the number of connections of node v in graph G and $|G|$ is the number of nodes in G. In its weighted version, the degree of a node is equal to the sum of the weights of its incident edges. 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:

$$\text{closeness}_G(v) = \frac{|G| - 1}{\sum_{v' \in V} dist(v, v')}$$

Where $dist(v, v')$ is the shortest path distance between nodes v and v' in G. Unlike degree centrality, closeness is a global metric, in that it combines information from all nodes in the graph.

2.2 Term Weighting-Inverse Collection Weighting (TW-ICW)

Instead of multiplying our graph-based component tw_p by the same term $idf(t, D) = \log\left({}^{m+1}/_{df(t)}\right)$, we can try to compute an IDF at the graph level. The goal is still the same: penalizing the terms that occur frequently at the collection level. To this purpose, we define the collection level graph G_D as the union[2] of all the graphs in the collection, i.e. $G_D = G_{d_1} \cup G_{d_2} \cup \ldots \cup G_{d_m}$. Note that instead of combining graphs constructed independently for each document, we can also obtain G_D by sliding a window over the entire collection while making sure that the window does not overspan documents. The inverse collection weight of a term t is then defined as:

$$icw(t, G_D) = \log\left(\frac{\max_{v \in G_D} tw(v, G_D) + 1}{tw(t, G_D)}\right)$$

Which gives the final TW-ICW measure as:

$$tw_p\text{–}icw(t, d) = \frac{tw(t, G_d)}{1 - b + b \times \frac{|d|}{avgdl}} \times icw(t, G_D)$$

TW-ICW, along with some other variants, were introduced by Skianis et al. [2018].

3 Coding time

Note: a list of igraph methods can be found here: `http://igraph.org/python/doc/igraph.Graph-class.html`. We will work with the WebKB dataset. It features academic webpages belonging to four different categories: (1) project, (2) course, (3) faculty, and (4) students, and contains 2,803 documents for training and 1,396 for testing. Documents have already been preprocessed with stopword removal and Porter's stemming. The code that you will work with implements the following steps:

- data loading,

[2] the union of two graphs G_1 and G_2 is defined as the union of their node and edge sets, i.e., $G_1 \cup G_2 = (V_{G_1} \cup V_{G_2}, E_{G_1} \cup E_{G_2})$

- computation of features (TF-IDF, TW-IDF and TW-ICW) for the training set,
- computation of features for the test set. Note that the documents in the test set are 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, the inverse document frequency, etc. should be the ones computed on the training set.
- classifier training/testing. Naive Bayes classifier (McCallum and Nigam [1998]), Multinomial Logistic Regression (Collins et al. [2002]) and linear kernel SVM (Joachims [1998]) are compared,
- the most (resp. least) important features for each class, are obtained as the highest (resp. lowest) weights of a classifier for each class.

Each of the steps above is implemented either entirely or partially in the `document_classification.py` script. The custom functions used can be found in the `library.py` file. As usual, **fill the gaps wherever needed**. You have 3 gaps to fill in `library.py` and 8 in `document_classification.py`. Questions[3]: Compare performance for different features and the same classifier, and for the same features but different classifiers. What are the best features/classifiers? 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.

Konstantinos Skianis, Fragkiskos Malliaros, and Michalis Vazirgiannis. Fusing document, collection and label graph-based representations with word embeddings for text classification. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pages 49–58, 2018.

[3] in-class only. You don't need to submit any response.