ADVANCED LEARNING FOR TEXT AND GRAPH DATA
MVA - MATHBIGDATA
Final Project: Text Categorization

Fragkiskos Malliaros and Michalis Vazirgiannis

March 24, 2015

1 Description of the Project

The goal of this project is to study and apply techniques for text categorization [5]. Text categorization (or text classification) is the task of assigning a document into a set of one or more predifined categories (i.e., classes). Such techniques are widely used in several domains, such as classification of news stories and webpages.

The pipeline that typically is followed to deal with the problem is similar to the one applied in any classification problem; the goal is to learn the parameters of a classifier from a collection of training documents (with known class information) and then to predict the class of unlabaled documents. The first step in text categorization is to transform documents, which typically are strings of characters, into a representation suitable for the learning algorithm and the classification task. Here, we will employ the *Vector Space Model*, a spatial representation of text documents. In this model, each document is represented by a vector in a n-dimensional space, where each dimension corresponds to a term (i.e., word) from the overall vocabulary of the given document collection. More formally, let $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$ denote a collection of m documents, and $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ be the dictionary, i.e., the set of words in the corpus $\mathcal{D}$. As we will descibe later, $\mathcal{T}$ is obtained by applying some standard natural language processing techniques, such as tokenization, stop-words removal and stemming. Each document $d_i \in \mathcal{D}$ is represented as a vector of term weights $d_i = \{w_{i,1}, w_{i,2}, \ldots, w_{i,n}\}$, where $w_{i,k}$ is the weight of term k in document d_i. That way, data can be represented by the *Document-Term matrix* $\mathbf{DT}$ of size $m \times n$, where the rows correspond to documents and the columns to the different terms (i.e., features) of set $\mathcal{T}$. Additionaly, each document d_i is associated with a class label y_i forming the class vector $\mathbf{Y}$, and the goal is to predict the class labels for a set of test documents. Based on this formulation, traditional classification algorithms (e.g., SVMs, Random Forests, etc.) can be applied to predict the category label of test documents.

The main issue with the Vector Space Model document representation is how to find appropriate *weights* for the terms. The goal of this project would be to explore the effect of two term weighting mechanisms in text categorization, that arise from two different representations of a document: (i) *bag-of-words* model and (ii) *graph-of-words* model. The former is the traditional document representation, where each document is represented as the bag (multiset) of its words, disregarding grammar and even word order. Under this model, the importance of a term for a document is mainly based on the frequency of this term. In the graph-of-words model, each document is modelled as a graph, whose vertices represent words and the edges capture relations between the words, defined on the basis of

any meaningful statistical or linguistic relation. As we will describe later, here we are interested for co-occurence relationships of two terms in a document within a window. That way, the importance of a term can be expressed using node centrality criteria, such as degree centrality.

In this project, we will use the *Reuters-21578* collection, that is composed by documents appeared on the Reuters newswire in 1987 (next, we will provide more details about the dataset), and we will examine the effect of the different representations in the text categorization problem.

2 Dataset Description and Preprocessing

The documents in the *Reuters-21578* collection appeared on the Reuters newswire in 1987 and were assembled and indexed with categories by personnel from Reuters Ltd. Many documents of this collection belong to multiple classes; here, we will consider only documents with a single category (class), and we will use a subset of the dataset called *Reuters-21578 R8* dataset. This dataset is split into the **Train**[1] part with $5,485$ documents and the **Test**[2] part with $2,189$ documents. Each of these documents belong to one of eight possible categories. Table 1 gives details about the dataset. The final evaluation of your methods will be done on the **Test** dataset and the goal will be to predict the class labels.

Class Label	# of Train docs	# of Test docs	Total # of docs
acq	$1,596$	696	$2,292$
crude	253	121	374
earn	$2,840$	$1,083$	$3,923$
grain	41	10	51
interest	190	81	271
money-fx	206	87	293
ship	108	36	144
trade	251	75	326
Total	$\mathbf{5,485}$	$\mathbf{2,189}$	$\mathbf{7,674}$

Table 1: Description of the *Reuters-21578 R8* dataset.

The datasets given here (both **Train** and **Test**) have already been preprocessed according to the following steps: (i) Substitute TAB, NEWLINE and RETURN characters by SPACE, (ii) Keep only letters (that is, turn punctuation, numbers, etc. into SPACES), (iii) Turn all letters to lowercase, (iv) Substitute multiple SPACES by a single SPACE. The two dataset files contain one document per line. Each document is represented by a word representing the document's class (according to the class labels given in Table 1), a TAB character and then a sequence of words delimited by spaces, representing the terms contained in the document. Furthermore, stopwords and words that are less than 3 characters long have been removed from the documents. Finally, Porter's stemmer has been applied, keeping only the root of each term. For example, the first document of the **Train** collection is shown in Fig. 1. The first word *earn* corresponds to the category of the document, and the document follows.

3 Document Representation and Term Weighting Mechanisms

As we discussed in Section 1, we will apply two document representation mechanisms and examine their effect to the categorization task. In both cases, the features in our classification problem correspond

[1]You can download the **Train** dataset from the following link: `http://web.ist.utl.pt/acardoso/datasets/r8-train-stemmed.txt`

[2]You can download the **Test** dataset from the following link: `http://web.ist.utl.pt/acardoso/datasets/r8-test-stemmed.txt`

```
earn champion product approv stock split champion product inc board director approv two for
stock split common share for sharehold record april compani board vote recommend sharehold
annual meet april increas author capit stock mln mln share reuter
```

Figure 1: Example of a preprocessed document from the collection.

to the different terms (words) of our collection; we are interested to find relevant weighting criteria for the Document-Term matrix, i.e., assign a relevance score to each term for each document. Next, we describe the basics of each representation.

3.1 Bag-of-Words Representation

This is the traditional document representation used in the Vector Space Model, and this will be the baseline approach for the experiments. In this case, the importance of a term $t \in \mathcal{T}$ within a document $d \in \mathcal{D}$ is based on the frequency $tf(t, d)$ of the term in the document (TF). Furthermore, terms that occur frequently in one document but rarely in the rest of the documents, are more likely to be relevant to the topic of the document. This is known as the inverse document frequency (IDF) factor, and is computed at the collection level. It is obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient, as follows:

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

where m is the total number of documents in collection $\mathcal{D}$, and the denominator captures the number of documents that term t appears. Having computed the TF and IDF metrics, we can combine them to get the TF-IDF score:

$$tf\text{-}idf(t, d, \mathcal{D}) = tf(t, d) \times idf(t, \mathcal{D}).$$

This function captures the intuitions that (i) the more often a term occurs in a document, the more it is representative of its content, and (ii) the more documents a term occurs in, the less discriminating it is[3]. Using the TF-IDF score of each term for each document, we can fill in the weights of the Document-Term matrix. As described in Ref. [5], page 13, in order to get normalized weights in the $[0, 1]$ interval, we can apply the cosine normalization.

3.2 Graph-of-Words Representation

The graph-of-words is a different document representation, that captures the relationships between the terms, questioning the term independence assumption [4, 1]. One of the main points that need to be specified is how to create a graph from the text collection. In general, each document $d \in \mathcal{D}$ is represented by a graph $G = (V, E)$, where the nodes correspond to the terms t of the document and the edges capture co-occurence relations between terms within a fixed-size sliding window of size w. That is, for all the terms that co-occur within the window, we add edges between the corresponding nodes (note that, the windows are overlapping starting from the first term of the document; at each step we simply remove the first term and add the new one from the document). Based on the graph representation, several points need to be specified:

- *Directed* vs. *undirected* graph. One parameter of this model is if the graph representation of the

[3]Several variants of the *tf-idf* score have been proposed. See also the description given in Ref. [5], pages 12-14.

document will be directed or undirected. Directed graphs are able to preserve actual flow of a text, while in undirected ones, an edge captures a co-occurrence of two terms whatever the respective order between them is. For example, in the field of Information Retrieval [4], directed graphs were selected to model documents. In this project, you need to examine which representation is more appropriate for text categorization.

- *Weighted* vs. *unweighted* graph. One approach is to consider the graphs as weighted. That is, the higher the number of co-occurences of two terms in the document, the higher the weight of the corresponding edge (i.e., the weight of each edge will be equal to the number of co-occurences of its endpoints). The second option is to consider the graphs as undirected. During the project, you should explore which of these two approaches perform better.

- Size w of the sliding window. As we described above, we add edges between the terms of the document that co-occur within a sliding window of size w. The size of the window is yet another parameter of the problem and should be explored in the project. Note that, in the Information Retrieval field, window of size $w = 4$ produced accurate results [4].

Term Weighting in the Graph-of-Words Representation

In the case where the document is represented using the bag-of-words approach, the term frequency criterion (or the TF-IDF score) constitutes the basis for weighting the terms of each document. How this can be done in the graph-of-words model? The answer is given by utilizing node centrality properties of the graph. That way, the importance of a term in a document can be extracted by the importance of the corresponding node in the graph. In graph theory and network analysis domains, several centrality criteria[4] have been proposed. Some of them have alredy been explored for graph-based Information Retrieval [4, 1] and keyword extraction [3]. Some of these measures consider only local information on the graph (e.g., degree centrality, indegree/outdegree centrality in directed networks, weighted degree in weighted graphs, clustering coefficient, etc.), while other are more global in the sense that the importance of a node is based on the properties of this node globally in the graph (e.g., PageRank centrality, eigenvector centrality, betweenness centrality, core number, etc.). The main goal of this project is to explore the capabilities of node centrality measures for text categorization (in contrast to the term frequency in the traditional bag-of-words representation). One important point here concerns the computational complexity of those centrality criteria. As we expect, some criteria are more efficient to be computed (e.g., degree centrality), while other no. This potential trade-off between classification accuracy and complexity of computing the features should also be explored.

In the case of TF-IDF metric, the frequency of each term in the document (TF factor) is penalized by the frequency of the term globally in the collection (IDF factor). Can we apply something similar here? One approach could be to ignore the corresponding factor from the term weighting process and simply keep the weight produced by the centrality measure per document (e.g., the degree of each term). Another approach could be to use graph features for the terms locally per document (e.g., degree) and penalize this term by the inverse document frequence (i.e., combination of graph and term frequency metrics – e.g., degree of a term per document (TD in a similar way to TF) penalized by the IDF factor). Lastly, one can derive something equivalent to the IDF factor (i.e., at the document collection level) for graph features (e.g., create a graph from all documents and consider centrality metrics (such as the degree) at the collection-level graph). The last point has not been explored in the related literature and it would be interesting to examine it in the context of the project.

[4]Wikipedia's lemma for *network centrality*: `http://en.wikipedia.org/wiki/Centrality`.

4 Summary of the Pipeline

Here we briefly describe each part of the pipeline that should be followed in the project.

Data preprocessing. As we described above, the data has already been preprocessed using standard tools and methods in the domain.

Dimensionality reduction. As we expect, the Document-Term matrix that will represent the data will have number of features proportional to the number of terms in the collection. Here, you can apply dimensionality reduction techniques to reduce the dimensionality of the dataset (see also Ref. [5], page 15). One approach is to apply traditional *feature selection* techniques (such as *Information Gain* or *Chi-square*), that in the text categorization problem can be interpeted as term selection (or term reduction). That way, the number of terms $|\mathcal{T}|$ is reduced to $|\mathcal{T}'|$, where $|\mathcal{T}| < |\mathcal{T}'|$. Other approach is to apply *Latent Semantic Indexing* (LSI), which actually performs SVD to the Document-Term matrix. That way, the data is projected into a lower dimensional space that can be used for the learning task.

Learning algorithm. The next step of the pipeline involves the selection of the appropriate learning (i.e., classification) algorithm for the problem (e.g., SVMs, Decision Tree, Random Forests). Here, you can test the performance of different algorithms and choose the best one. For example, it has been shown that SVMs perform better in text categorization compared to other classification algorithms [2]. Additionally, you can follow an ensemble learning approach[5], combining many classification algorithms (e.g., *AdaBoost* algorithm[6]).

Evaluation. You will build your classification model based on the **Train** dataset. To do this, you can apply *cross-validation* techniques[7]. The goal of cross-validation is to define a dataset to test the model in the training phase, in order to limit problems like overfitting and have an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, like the **Test** dataset that will be used to assess your model). Of course, having a good model that achieves good accuracy under cross validation does not guarantee that the same accuracy will also be achieved for the test data. Thus, the final evaluation of your model will be done on the test dataset contained in the **Test** dataset; after having a model that performs well under cross-validation, you should train the model using the whole **Train** dataset and test it on the **Test** dataset.

The performance of the classifier will be assessed using the measures of *precision* and *recall* with respect to the different categories. That is, the classification effectiveness is measured per category (i.e., class) and then is averaged over all categories to have the final result. For each different class $c_i, \forall i = 1, \ldots, k$ (k is the total number of classes), the contingency table is computed, i.e., the following quantities: FP_i (false positives *wrt* c_i) is the number of test documents incorrectly classified under c_i, TN_i (true negatives *wrt* c_i), TP_i (true positives *wrt* c_i) and FN_i (false negatives *wrt* c_i). Then, the precision $\hat{\pi}_i$ and recall $\hat{\rho}_i$ with respect to class c_i are estimated:

$$\hat{\pi}_i = \frac{TP_i}{TP_i + FP_i} \qquad \text{and} \qquad \hat{\rho}_i = \frac{TP_i}{TP_i + FN_i}.$$

Then, the total precision and recall values can be estimated by the (i) *microaveraging* and (ii) *macroaveraging* methods over all k different categories:

[5]Wikipedia's lemma for *Ensemble learning*: `http://en.wikipedia.org/wiki/Ensemble_learning`.
[6]Wikipedia's lemma for *AdaBoost*: `http://en.wikipedia.org/wiki/AdaBoost`.
[7]Wikipedia's lemma for *Cross-validation*: `http://en.wikipedia.org/wiki/Cross-validation_(statistics)`.

- *Microaveraging*: the precision and recall values are obtained by summing over all individual decisions:

$$\pi^{\mu} = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} (TP_i + FP_i)} \quad \text{and} \quad \rho^{\mu} = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} (TP_i + FN_i)}.$$

- *Macroaveraging*: the precision and recall are first evaluated locally for each category, and then globally by averaging over the results of the different categories:

$$\pi^{M} = \frac{\sum_{i=1}^{k} \hat{\pi}_i}{k} \quad \text{and} \quad \rho^{M} = \frac{\sum_{i=1}^{k} \hat{\rho}_i}{k}.$$

In the project, both these measures should be used for the evaluation of the classification task. For a more detailed description, please see Ref. [5], pages 37-38, this link[8], and the experimental results of Ref. [2].

Additionally, depending on the graph features that you will choose from the graph-of-words representation, the overall running time of your approach will be affected (more precisely, the feature extraction step). For example, the PageRank score is computationally more intensive than the degree centrality. We expect to take into account this factor during the evaluation of your approach and to examine trade-offs between the time needed for feature extraction and accuracy results.

5 Useful Python Libraries

In this section, we briefly discuss some tools that can be useful in the project and you are encouraged to use.

- A very powerful machine learning library in Python is `scikit-learn`[9]. It can be used in the preprocessing step (e.g., for feature selection) and in the classification task (a plethora of classification algorithms have been implemented in `scikit-learn`). Also, in `scikit-learn` you will find built-in functions for the microaverage and macroaverage precision and recall metrics.

- Since you will deal with textual data, the Natural Language Toolkit (`NLTK`)[10] of Python can also be found useful.

6 Details about the Submission of the Project

The project can be done in **teams of 2-3 people**. Your final evaluation for the project will be based on both the precision and recall values that will be achieved on the **Test** dataset, as well as on your total approach to the problem. As part of the project, you have to submit the following:

- A 2-3 pages report, in which you should describe the approach and the methods that you used in the project. Since this is a real classification task, we are interested to know how you dealt with each part of the pipeline, e.g., how you created the graph-of-words representation, which graph features did you use, if you applied dimensionality reduction techniques, which classification algorithms did you use and why, the performance of your methods (accuracy and training time),

[8] `http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-text-classification-1.html`
[9] `http://scikit-learn.org/`
[10] `http://www.nltk.org/`

approaches that finally didn't work but is interesting to present them, and in general, whatever you think that is interesting to report. Also, in the report, please provide the names of the team members.

- A directory with the code of your implementation.
- Create a `.zip` file with name `ALTEGRAD_Ex2_Lastname1_Lastname2.zip` (the lastnames of all team members), containing the code and the report and submit it to moodle.
- **Deadline: Sunday, April 19, 23:55.**

References

[1] Roi Blanco and Christina Lioma. Graph-based term weighting for information retrieval. *Inf. Retr.*, 15(1):54–92, February 2012.

[2] 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, 1998. Springer-Verlag.

[3] Shibamouli Lahiri, Sagnik Ray Choudhury, and Cornelia Caragea. Keyword and keyphrase extraction using centrality measures on collocation networks. `http://arxiv.org/pdf/1401.6571v1`, 2014.

[4] 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 and Knowledge Management*, CIKM '13, pages 59–68, New York, NY, USA, 2013. ACM.

[5] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.