Unsupervised Document Classification and Topic Detection

Jaromír Novotný(✉) and Pavel Ircing

Faculty of Applied Sciences, Cybernetics, The University of West Bohemia, Plzeň, Czech Republic
{fallout7,ircing}@kky.zcu.cz
http://www.kky.zcu.cz/en

Abstract. This article presents a method for pre-processing the feature vectors representing text documents that are consequently classified using unsupervised methods. The main goal is to show that state-of-the-art classification methods can be improved by a certain data preparation process. The first method is a standard K-means clustering and the second Latent Dirichlet allocation (LDA) method. Both are widely used in text processing. The mentioned algorithms are applied to two data sets in two different languages. First of them, the 20NewsGroup is a widely used benchmark for classification of English documents. The second set was selected from the large body of Czech news articles and was used mainly to compare the performance of the tested methods also for the case of less frequently studied language. Furthermore, the unsupervised methods are also compared with the supervised ones in order to (in some sense) ascertain the upper-bound of the task.

Keywords: Text pre-processing · Classification · Evaluation · LDA · K-means

1 Introduction

This work deals with the preparation of input text data and consequent document classification using unsupervised methods.

Since a significant portion of the algorithms used for document classification internally utilizes some measures of vector similarity, one of the crucial steps of document pre-processing is the conversion of input text into some kind of a vector representation. The basic approach to such conversion is a so-called Bag-of-Words model(BOW) [4] – in such case, each document is represented by a vector where each element corresponds to a word from a fixed position in the lexicon. The value of such element is usually directly proportional to the number of occurrences of the given word in the given document (term frequency – tf) and indirectly proportional to the number of documents where the given word occurs (the inverse document frequency – idf). The resulting tf-idf model is very successful [2,6,7]. However, sometimes the length and sparseness of the resulting

A. Karpov et al. (Eds.): SPECOM 2017, LNAI 10458, pp. 748–756, 2017.
DOI: 10.1007/978-3-319-66429-3_75

vector, stemming from the size of the lexicon, may hurt the performance of the classification algorithms. Several methods for the reduction of the vector dimension are therefore discussed later and constitute the core of our work.

For the classification itself, we have picked two methods – the "classic" clustering algorithm K-means, which is simple but is known to perform well if we are able to present it with the suitable feature vectors, and the state-of-the-art methods for unsupervised topic detection, the Latent Dirichlet allocation (LDA), adapted for document classification.

2 Datasets

As our basic dataset, we have picked the *20NewsGroups* English corpus[1] which is widely used as a benchmark for document classification [7,9,12,13]. It contains 20 000 text documents which are evenly divided into 20 categories that each contain discussion about a specific topic. The second data set *CNO* is in Czech language and contains also approximately 20 000 articles divided into 31 categories[2]. This corpus was created so that it is at least in size and partially also in topics comparable to the English data set.

In order to compare our results with the ones published previously, we have re-created two subdivisions of the *20NewsGroups* corpus. The first one is created according to [13,14] and consists of the following subsets:

- Set *20NG* consists of all 20 original categories but includes only documents containing at least 10 word tokens (after stop-word removal). This results in approximately 17 000 documents in total.
- Set *10NG* consists of the same documents as the *20NG* above but divides them into 10 categories only – the reduced number of categories was obtained by merging 5 original `comp`, 3 `religion`, 3 `politics`, 2 `sport` and 2 `transportation` categories into one category for each "domain".
- The next group of subsets contains 9 sets for small-scale experiments – there are three *Binary*, three *5Multi* and three *10Multi* sets, each containing 500 documents only and prepared in the following way:
  - Binary subsets (denoted *Binary[0/1/2]*) are created by randomly choosing 2 categories (from the original 20) and randomly drawing 250 documents from each of them.
  - Analogically, the *5Multi[0/1/2]* subsets were created by randomly choosing 5 original categories and randomly drawing 100 documents from each.
  - And finally, the *10Multi[0/1/2]* subsets were created by randomly choosing 10 original categories and randomly drawing 50 documents from each.

[1] This data set can be found at http://qwone.com/~jason/20Newsgroups/ and it was originally collected by Ken Lang.

[2] It was created from a database of news articles downloaded from the http://www.ceskenoviny.cz/ at the University of West Bohemia and constitutes only a small fraction of the entire database – the description of the full database can be found in [16].

The other subdivision is created in order to compare the results with experiments described in [12]. This set, denoted as *Binary20NG*, is comprised of 20 bi-classes – each bi-class consists of one class containing all the documents from one of the original categories (i.e., 1000 documents) and the second class containing 1000 documents randomly drawn from the pool of other 19 categories. Two-thirds of each such bi-class documents are used as the training data, the remaining third constitutes the test set.

The *CNO* set was not subdivided in any such way.

3 Preprocessing

First, we removed all the headers from the *20NewsGroups* data, except for the `Subject`. Then all uppercase characters were lower-cased and all digits were replaced by one universal symbol.

As the next processing step, we wanted to conflate different morphological forms of the given word into one representation. This can be achieved by either lemmatization or stemming – even though those two procedures have rather similar outputs, we opted for lemmatization. The MorphoDiTa [15] tool was picked for the task – it works for both English and Czech and is available as a Python package.[3]

Further preprocessing traditionally comprises stop-word removal.[4] Probably the most common approach is to use a pre-defined stoplist, but the stop words can also be determined on the basis of the input data analysis. We use a simple method of detecting stop words from input data. We compute for each lemma its inverse document frequency [6] (*idf*):

$$idf_l = \frac{N}{N(l)} \tag{1}$$

where N is a total number of documents and $N(l)$ denotes a number of documents containing the lemma l. Then we set a threshold θ and classify every lemma l with $idf_l < \theta$ as a stop word and remove it from further processing.

At this point, we have a suitable data for the LDA analysis as it starts from the set of preprocessed documents (see the details in Sect. 4.1).

However, more data processing is needed when preparing input for the K-means algorithm. We need to compute the tf-idf weights $w_{l,d}$ for the lemmas $l \in L$ and documents $d \in D$ using the well-known formula:

$$w_{l,d} = tf_{l,d} * idf_l \tag{2}$$

where $tf_{l,d}$ denotes the number of times the lemma l occurs in document d and idf_l is computed using the Eg. (1).

[3] `ufal.morphodita` at https://pypi.python.org/pypi/ufal.morphodita.

[4] The authors of the paper [12] don't use the stop words at all because their feature vector consists only of the top T tokens (lemmas or stems) with highest mutual information (MI).

Besides the basic equations above, there are actually more sophisticated formulas for computing tf and idf available [6]. Many of them are implemented in the Python package `sklearn` [10][5] that we extensively use in essentially all further experiments. The `TfidfVectorizer` takes the set of input documents (preprocessed as described above) and (optionally) a dictionary and outputs the $|D| \times |M|$ matrix, where $|D|$ is the number of documents and $|M|$ is the number of features representing each document (in our case it is of course only the number of distinct lemmas occurring in all the preprocessed documents – L – but the feature set can be much richer – e.g. it can include also the higher order n-grams).

This matrix can be used as input for K-means method directly but it is usually beneficial to lower the dimension $|M|$ in order to lower the computational costs of the algorithm. We have decided to reduce the feature vector dimension using the well-know Latent Semantic Analysis (LSA) [5] which does not only lower the vector dimension but allegedly also captures some of the semantics hidden in the documents. The LSA method is again implemented in the Python package `sklearn` – the concerned module `TruncatedSVD` takes the input $|D| \times |M|$ matrix and produced a $|D| \times |R|$ matrix ($|R|$ being the desired lower dimension passed as a function parameter) that can be consequently used as an input for the K-means method.

4 Classification Methods

4.1 LDA

Latent Dirichlet allocation (LDA) is a generative probabilistic model of a corpus [1]. Marginally, documents are represented as random mixtures over latent topics (the latent multinomial variables in the LDA model are referred as topics). Each topic is then characterised by a distribution over terms (in our case, lemmas).

The LDA model itself and the related data preparation functions are implemented in the Python package `gensim` [11]. The documents preprocessed as described in Sect. 3 are first converted to special gensim's bag-of-words representation called *corpus* using the `doc2bow` function and the special *dictionary* file is also created.

The LDA method itself then uses both the *dictionary* and the *corpus* as its input; the model finds the list of topics (number of topics matches the number of categories of input data) that fits the input data. We set model to classify every document into one cluster only, that is, only the topic with highest probability is a assigned to each document. This model is applied on all prepared corpora (*20NG*, *10NG*, *Binary[0/1/2]*, *5Multi[0/1/2]*, *10Multi[0/1/2]*, *Binary20NG*, *CNO*) and results can be found in Sect. 6.

[5] More precisely the `TfidfVectorizer` module from that package.

4.2 K-Means

The classic K-means clustering method [8] is being used here as a classification algorithm. It is generally accepted that even such a simple method is quite powerful for unsupervised data clustering if it is given an appropriate feature vectors. Since we had good reasons to believe that our feature vectors consisting of the tf-idf weights capture the content of the document rather well (and the reduced feature vectors obtained from LSA do it even better), we expected to find the documents with similar topic often in only one of the clusters discovered by K-means.

We have used the version of K-means algorithm implemented in our favorite `sklearn` package. First, we used the full matrix of tf-idf weights; however, given the large dimension of such feature vectors, the clustering was feasible only for a small subset of the documents. The full experiments were performed with the reduced feature vectors obtained by applying the LSA. Again, we applied this model on all the date sets described in Sect. 2 and results can be found in Sect. 6.

5 Evaluation

There are quite a few measures for evaluation of the classification algorithms. In our experiments, we have decided to use accuracy, precision and recall; this choice was guided mostly by the fact that we wanted to compare the performance of our algorithms to the previously published results.

The *Accuracy* measure is applied on *Binary20NG* data set and represents the percentage of correctly classified documents (i.e., show what percentage of the test documents is assigned with the correct topic).

Precision and *Recall* measures are computed according to [13] and are used on data sets *20NG*, *10NG*, *Binary[0/1/2]*, *5Multi[0/1/2]*, *10Multi[0/1/2]* and *CNO*.The micro-average type of those measures is applied. One dominant category $c \in C$ is assigned to all output clusters $t \in T$. This is done by computing number of documents which are the same in t and c, the highest value then designate the dominant category c to cluster (output) t. Every $c \in C$ can be assigned only to one $t \in T$. This procedure is done in [13], because of the underlying assumption that user would not have a problem with assigning a dominant topic (if the clusters are relatively homogeneous). We can then define the following quantities: $\alpha(c, T)$ which defines the number of documents correctly assigned to c, $\beta(c, T)$ defines the number of documents incorrectly assigned to c and $\gamma(c, T)$ defines the number of document incorrectly not assigned to c. It is now possible (from those values) to compute micro-average precision P and recall R as follows:

$$P(T) = \frac{\sum_c \alpha(c, T)}{\sum_c \alpha(c, T) + \beta(c, T)} \tag{3}$$

$$R(T) = \frac{\sum_c \alpha(c, T)}{\sum_c \alpha(c, T) + \gamma(c, T)} \tag{4}$$

Since the original corpus and output from algorithms are uni-labeled data sets in this scenario, the $P(T)$ is necessarily equal to $R(T)$ (number of original categories in corpus have to be also the same as the number of output clusters from algorithms) and it is sufficient to report only one of those values. That's why there is only *Precision* reported in Table 2.

6 Results

First, we report the average results achieved on $Binary20NG$ data set in Table 1. This set of results is compared with results of [12]. The mentioned paper employs supervised methods, two of them baselines and the others are those baseline methods improved by semantic smoothing of the kernels. First used method is K-nearest neighbors (denoted by *KNN*) and the corresponding method with smoothed kernel is denoted by *KNN+P*. The second method is support vector machines (*SVM*) and the corresponding method with smoothed kernel is *SVM+P*.

The results of our methods are listed in the lower part of the table and denoted as *LDA* and *K-means*. The K-means method was applied with both the full matrix of tf-idf weights and with the matrix reduced by LSA (reduced feature vectors of size 2). Sice the work reported in [12] concerns supervised training, the data set in question was designed to consist of training and test portion. We have preserved the partitioning but naturally did not use the supervisory information from the training part for our unsupervised methods.

Table 1. Comparison of our results with results achieved in [12]

Method	*Accuracy* [%]
KNN	71.79
KNN+P	80.13
SVM	86.44
SVM+P	88.52
LDA	56.46
K-Means (tf-idf matrix from all lemmas)	72.19
K-Means (matrix form LSA method, number of features is 2)	75.47

Second sets of results are listed in Table 2; these results were achieved on *20NG*, *10NG*, *Binary[0/1/2]*, *5Multi[0/1/2]*, *10Multi[0/1/2]* data sets. Again, we are comparing our results with the values reported in the previously published paper, this time [13]. The authors of the mentioned paper used the (unsupervised) *sIB* and *sK-means* methods. The *sIB* stands for sequential Information Bottleneck method and the *sK-means* stands for sequential K-means method (modification of the K-means). This modification lies in updating the centers of

the clusters whenever a feature vector is assigned to one of them (not at the end – after all of the feature vectors are assigned – as in classical K-means algorithm). In our experiments, we run our LDA and K-means algorithms 10 times over each subset (same approach used in [13]). Averaged results from those runs are listed in Table 2. The meaning of the K-means experiment labels listed in the table is the following:

- *K-means* is the algorithm run with full tf-idf weights
- *K-means(LSA)* is the algorithm run with feature vectors reduced by LSA to the dimension equal to the number of original categories (except for sub-set *20NG*, where the number of features is set on 2000)
- *K-means (LSA n features)* states results from K-means method with input matrix produced by LSA method, which lowers dimension to $n = 144$ for large data sets (*20NG* and *10NG*) and to $n = 46$ for small data sets (the rest of data sets in Table 2). The values of n for data sets were computed by using formula listed in [3], which is:

$$n = n_T^{\frac{1}{1+\frac{\log(n_T)}{10}}} \tag{5}$$

where n_T is number of texts (documents).

Table 2. Comparison of our results with results achieved in [13]

20NewsGroups sub-sets	*Precision* of methods [%]					
	sIB	*sK-means*	*LDA*	*K-means*	*K-means (LSA)*	*K-means (LSA n features)*
20NG	57.50	54.10	16.97		38.08	35.81
10NG	79.50	76.30	28.72		45.29	51.04
Average "large"	68.50	65.20	22.84		41.69	43.43
10Multi0	70.20	31.00	30.40	36.68	49.98	51.40
10Multi1	63.80	32.80	23.80	36.72	49.88	52.92
10Multi2	67.00	32.80	32.59	45.22	60.12	63.00
5Multi0	89.40	47.00	43.00	71.76	70.06	76.54
5Multi1	91.20	47.00	46.20	73.50	79.80	84.58
5Multi2	94.20	57.00	36.20	68.64	73.00	79.12
Binary0	91.40	62.40	95.60	94.50	98.80	97.60
Binary1	89.20	54.60	94.00	92.64	93.80	92.90
Binary2	93.00	63.20	93.38	97.48	97.20	97.48
Average "small"	83.30	47.60	55.02	68.57	74.74	77.28

Finally, we list some results from *CNO* data set in Table 3. These are only for the purpose of testing our methods on data in different language than English. This result shows that our approach to the preparation of data can be applied even for the language rather distant from English. We tested different settings on lowering dimension with LSA method. First was set to 2000 and second to 31 (which corresponds to a number of original categories).

Table 3. Results on *CNO* data set

	Precision of methods [%]		
	LDA	K-means (dim. reduced to 2000)	K-means (dim. reduced to 31)
CNO	14.67	42.59	41.72

7 Conclusion

The paper introduced a reasonably effective pipeline for classification of the text documents according their topic. It concentrated mostly on the preprocessing of both the raw input text and the extracted feature vectors. It showed that when applying lemmatization and data-driven stop-word removal to the text documents and consequently reducing the dimension of resulting tf-idf feature vector using LSA, we can get decent classification results even with the most rudimentary classification algorithms, such as K-means. The performance of this unsupervised method was almost on par with some of the simpler supervised algorithms. This is an important finding of our research, since the training data annotated with correct document classification – which are necessary for supervised learning – are often not available.

Acknowledgements. This research was supported by the Ministry of Culture of the Czech Republic, project No. DG16P02B048.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
2. Eklund, J.: With or without context: automatic text categorization using semantic kernels. Ph.D. thesis, University of Borås, Faculty of Librarianship, Information, Education and IT (2016)
3. Fernandes, J., Artífice, A., Fonseca, M.J.: Automatic estimation of the LSA dimension. In: Proceedings of the International Conference on Knowledge Discovery and Information Retrieval (IC3K 2011), pp. 301–305 (2011)
4. Huang, A.: Similarity measures for text document clustering. In: Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand, pp. 49–56 (2008)

5. Landauer, T.K., Foltz, P.W., Laham, D.: An introduction to latent semantic analysis. Discourse Process. **25**, 259–284 (1998)
6. Lehečka, J., Švec, J.: Improving multi-label document classification of Czech news articles. In: Král, P., Matoušek, V. (eds.) TSD 2015. LNCS (LNAI), vol. 9302, pp. 307–315. Springer, Cham (2015). doi:10.1007/978-3-319-24033-6_35
7. Liu, Y., Liu, Z., Chua, T.S., Sun, M.: Topical word embeddings. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 2418–2424 (2015)
8. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: 5-th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
9. Nguyen, D.Q., Billingsley, R., Du, L., Johnson, M.: Improving topic models with latent feature word representations. Trans. Assoc. Comput. Linguist. **3**, 299–313 (2015)
10. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011). http://scikit-learn.org
11. Řehůřek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50 (2010). https://radimrehurek.com/gensim/
12. Siolas, G., d'Alche Buc, F.: Support vector machines based on a semantic kernel for text categorization. In: IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN), vol. 5, pp. 205–209 (2000)
13. Slonim, N., Friedman, N., Tishby, N.: Unsupervised document classification using sequential information maximization. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 129–136 (2002)
14. Slonim, N., Tishby, N.: Document clustering using word clusters via the information bottleneck method. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 208–215 (2000)
15. Straková, J., Straka, M., Hajič, J.: Open-source tools for morphology, lemmatization, POS tagging and named entity recognition. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 13–18 (2014)
16. Švec, J., Lehečka, J., Ircing, P., Skorkovská, L., Pražák, A., Vavruška, J., Stanislav, P., Hoidekr, J.: General framework for mining, processing and storing large amounts of electronic texts for language modeling purposes. Lang. Resour. Eval. **48**(2), 227–248 (2014)