Classifying Reuters-7083 dataset. A Multinomial Naive Bayes approach

Author: Ioan-Bogdan Canciu

Abstract

Over the past few years, natural language processing is the branch of artificial intelligence that got more and more researchers hooked in. Reuters provides a large Corpus of articles that are being used in order to conduct researches and development of natural language processing(NLP) and machine learning systems. Previous approaches that have proven to return the best results vary from using word embedding regularization together with a vector space model representation to more complex message passing attention networks that are able to understand and classify documents of this Corpus. The goal of the current paper is to develop a system using a few well known preprocessing techniques together with a Multinomial Naive Bayes algorithm in order to be able to classify the documents of Reuters-7083 dataset. By the end of this article, our goal is to offer a good understanding on how the current approach was implemented, starting from feature extraction and going through all the steps until reaching the feature selection and the usage of the probabilistic learning method, Multinomial Naive Bayes.

1 Introduction

The main reason this research was conducted is to present a new approach in text mining and documents classification, going through multiple preprocessing phases and ilustrating each of the obtained results. Similar researches have been published during the past 3 years, some of them having unexpected great results. The most recent research of Vit Novotny et al.[1] describes how using word embedding regularization together with a vector space model representation returns a prediction accuracy of 92.65 on the Reuters Corpus. The results were not the best at that time, but they were providing a new perspective on how you could build a text classification system. One year before the Novotny publication, Giannis Nikolentzos et al. published a paper[2] that implements a whole new approach for that time. Starting from the graph neural networks, their work was focused on developing a message passing attention network for documents understanding which has the highest prediction accuracy that has been seen in the last few years, 97.44.

In our work, we investigate how impactful different feature extraction and feature selection methods are and we present our obtained results in a step-by-step manner. We show how the data is represented in order for a better understanding on how the future steps of the algorithm work and we will provide a refference to the Python libraries used for the Multinomial Naive Bayes algorithm.

The rest of the paper is organized as follows: We present the whole preprocessing phase together with the feature selection method in the Section 2. In Section 3, we discuss the way our data is represented and how we will fit it to the learning algorithm together with the usage of different Python modules in order to implement the NB(Naive Bayes). Finally, in the Section 4 we will conclude the paper and we will compare our results with already existing results obtained by other researchers.

2 Preprocessing and Feature Selection

The preprocessing steps were selected based on some of the multiple approaches that were discused by Dr. Vijayarani Mohan et al. on her preprocessing techniques paper[3]. The first step is represented by parsing the whole dataset, extracting and counting all the words from <title> and <text> tags. For each document of the dataset, a local dictionary will be used in order to store all distinct words together with the number of occurence each word has. Once all local dictionaries are created, a global dictionary containing each distinct word from each document, together with the grand total number of occurence for each word will be created. The topic of each document is also extracted in this step, in order to be able, later on, to train our learning algorithm.

The next preprocessing step consists of eliminating the words that are in a stopword list and removing the possessives from the nouns. Then, we will convert all characters to lower case and remove numbers and words that contain numbers. The punctuation marks will also be removed, and words that contains special characters such as „@" which is usually used to describe an e-mail address will also be removed. Now, that most of our words represents real features that can be used later on in training phase of the NB algorithm, we will extract the stem of each word in order to drop using multiple variations of the same word, keeping only the stem of it.

After all the preprocessing steps described so far, there is a total of 23247 distinctive words in the global dictionary, which is quite a lot. Some of the documents might be irrelevant for our research, so the next step will be figuring out which documents could be dropped without losing relevant information.

Our purpose is to build a classifying system, which could be able to predict what's the most suited class for a specific document. In order to do so, we must find out what classes are the most relevant for our classifier, this way we will be able to drop the classes that are less relevant. Based on the multiple simulations result, the test cases show the classes that occur on more than 95% of the documents, or in less than 5% of the documents, are irrelevant and can be dropped. This way we will filter the classes, keeping only the most relevant ones. After filtering the classes, some of the documents end up with an empty class list, which means such documents can be dropped.

Table 1. Illustration of classes filtering methods.

Filtering method	**Dataset documents**	**Number of classes**	**Number of words**
no filter	7083	68	23247
5-95 rule	6826	15	22432
10-90 rule	4294	5	12955

The results presented in Table 1 shows the impact of our current 5-95 rule(drop classes that occurs in more than 95% of documents or less than 5%) compared with a 10-90 rule. We can notice that, before applying this filter, we had 7083 documents in the dataset with a total of 68 different classes. The 5-95 rule slightly impacts the number of documents and number of words, with a decrease of about 3.65% compared with the whole dataset. On the other hand, a huge impact can be seen in terms of number of classes. Over 75% of the total number of classes has been dropped, which is a great result. The 10-90 rule overfilters the dataset, dropping around 40% of the documents and 93% of classes. This method shows an unequal tradeoff, which is why the 5-95 rule is the one we stick with.

Now that we managed to drop a few classes together with remaining unclassified documents, the next step would be finding a way to compute which is the most suited class for each document. At this point, each of the remaining documents belongs to one or more classes. Working with variable length of topics list is insubstantial, which is why we will define a method that assigns each document a single class. The method described by our classifying system will check each of the 15 remaining classes and will rank them based on their occurence. This way, each document will end up belonging to a single class, the one that's the most relevant out of all. After applying this method, a total of 9 classes were considered as being the most impactful ones, so the other 6 were dropped.

The next step consists of finding a way to filter the number of words(attribues). This step is defined as feature selection. The information gain is the method we chose for finding out which attributes worth keeping or not. In order to compute the Information Gain, we started by calculating the the Entropy of each data set. The Entropy is defined as:

$$Entropy(D) = \sum_{i=1}^{c} p_i log_2(p_i) \quad (1)$$

where D is a collection of n documents grouped into c classes, p_i is the proportion of elements of D belonging to class i.

Now that we defined the entropy, we will continue with presenting the way our feature selection mechanism works. Information Gain is the method we used to calculate the relevance of each attribute. The Information Gain formula was taken from Daniel I. Morariu et al. paper[4] and it is defined as:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \quad (2)$$

where *Values(A)* is the set of all possible values for attribute A, and Sv is the subset of S for which attribute A is equal to v.

In order to be able to retrieve valuable information using Information Gain method, we decided to label each attribute as it follows: 0 – if an attribute doesn't occur in a document, 1 – if an attributed occurs below average and 2 – if an attribute occurs above average. Before representing our data like this, we had the local dictionary of each document form the data set represented as a key-value pair where key was attribute and pair representic the occurence of the attribute in the current document. We used the global dictionary to compute the average occurence of an attribute in the whole dataset. The result of this averaging shows the average occurence of an attribute considering the whole dataset is around 31 times. Using this information, we reformatted our data as presented above, labeling each attribute with a value of 0, 1 or 2.

After computing the Information Gain, several tests has been done in order to decide what's the percentage of relevant attributes that can be used later on to train our learning algorithm. Starting from the approach presented by Sofie Van Landeghem et al. in their paper[5], we gradually decreased the amount of information we decided to keep, starting from keeping the most significant 25% of the attributes and going down to keeping only 10% of it.

Table 2. Prediction accuracy based on different Information Gain thresholds.

Amount of attributes	**Prediction accuracy**	**Number of words**
25%	0.782714	5608
20%	0.789132	4486
15%	0.801966	3365
10%	0.803719	2243

As Table 2 shows, our algorithm average prediction accuracy is reported as being the best when keeping only 10% of the most significant attributes. More than that, the time it takes for the algorithm to train also decreases with the number of kept attributes, which is even better. This way, less time it takes to train in order to be able to do better predictions and this is why we decided to stick with keeping only 10% of the most significant attributes and drop the others.

3 Dataset representation and Multinomial Naive Bayes algorithm

Now that we described the whole preprocessings steps we've been through, we will present the way our dataset is represented and how we will fit it to the NB algorithm. As we've already seen above, we had to format the data in a manner that the Information Gain algorithm can take advantage of. Labeling each attribute based on its occurence doesn't positively affect the learning algorithm, so we will move back to the dictionaries containing key-value pair, where the value represents how many times an attribute appears in a specific document.

At this point, our global dictionary holds only the most significant attributes that were filtered using the Information Gain method. The next step consists of updating each of the local dictionaries in order to remove unneccesary attributes, keeping only the ones that occur in the global dictionary. Once each local dictionary is updated, the final dataset representation will be done.

Table 3. Final dataset representation.

Document	**usa**	**ceo**	**comput**	**product**	**overcompl**	**…**	**industri**
100034NE.XML	1	1	8	11	4	…	3
100297NE.XML	5	3	1	12	0	…	4
100385NE.XML	1	0	4	0	0	…	0
100415NE.XML	0	6	2	2	12	…	3
⋮							
99792NEW.XML	1	1	0	0	0	…	0

Table 3 shows how our final dataset is represented, as a table. In reality, our dataset is stored as a *.csv* file representing what's the occurence of each attribute considering a specific document. The dataset representation phase is meant to serialize the documents attributes in a manner that we're be able later on to retrieve the information from, this is way we decided to choose *.csv* files.

As we've discused in the first section of the current paper, our approach relies on developing a text mining system using the Python programming language, together with a few modules that has already been developed by other researchers. In order to be able to fit our data to the training algorithm, we have to deserialize it from the *.csv* file and convert it into a special Python array type, NumPy.

Definition 1 NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.[6]

Our multidimensional array is meant to store the data as shown in the Table 3, where the column index represents a given attribute and the row index represents a given document.

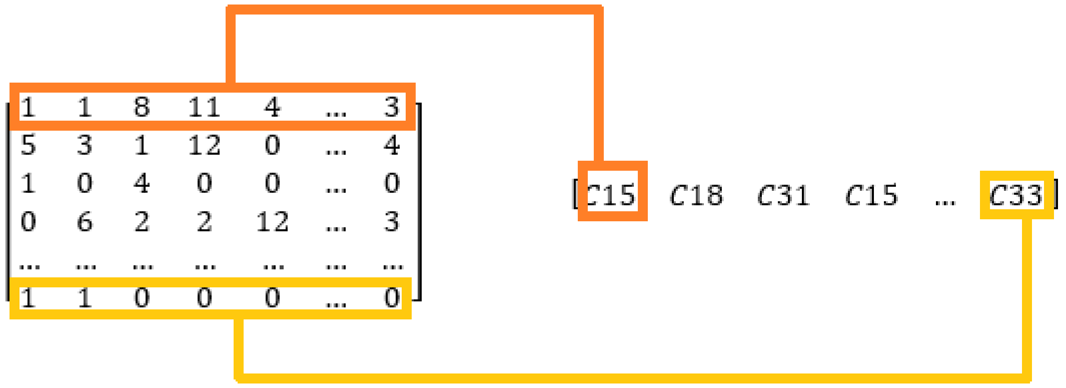
Fig 1. Dataset represented as a matrix.

$$\begin{bmatrix} 1 & 1 & 8 & 11 & 4 & \dots & 3 \\ 5 & 3 & 1 & 12 & 0 & \dots & 4 \\ 1 & 0 & 4 & 0 & 0 & \dots & 0 \\ 0 & 6 & 2 & 2 & 12 & \dots & 3 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

Figure 1 shows how our final dataset is represented as a NumPy matrix. Comparing the matrix with the previous table, Table 3, we notice that each attribute occurence is extracted from each document in order to be able to keep our data in a data structure that can be used later on to train the NB algorithm.

Now that we know how our data looks like, it’s time to move forward and talk about how the results are stored. The results represent the classes each document is classified into. In a similar manner, the classes are stored in a one-dimension array, each index representing the class for the exact specific row in the matrix.

Fig 2. Correlation between dataset attributes and classes as data structures.



For a better understanding on how our data structures are correlated, Figure 2 explains how each row of the attributes matrix is correlated to each index of the one-dimensional array that stores the document’s class. The reason we had to represent our data as shown above is to be able to fit it into our classification algorithm. The Multinomial Naive Bayes algorithm was implemented using a Python module called Scikit-learn.

Definition 2 Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection, model evaluation, and many other utilities. [7]

The *fit* method of the algorithm takes 2 arguments: first one is the training matrix, which represents an arbitrary number of documents out of the whole dataset, and the second one is the classes array, which represents an arbitrary number of classes based on the documents we choose to train our algorithm with.

4 Conclusions

In this paper we have evaluated the main preprocessing steps of building a document classification system together with Information Gain feature selection method and we’ve seen how Multinomial

Naive Bayes algorithm performs. Our experiments were done using quite a few Python open source modules, that were extremely useful for our research from both time-efficiency perspective and algorithm complexity perspective.

Speaking of the experimental results and comparing them with other results obtained during the past few years, several tests has been done using the Scikit-learn module. For the current paper, we chose to split the dataset in training data and learning data as follows: 70% of the dataset was used to train the algorithm and 30% of the dataset was used to test it.

Table 4. Comparison in terms of prediction accuracy.

Model	**Prediction accuracy**
Multinomial NB	80.37%
Soft VSM	92.65%
REL-RWMD kNN	95.61%
MPAD-path	97.44%

Table 4 shows how the current approach comapers with others that were published in the past 3 years. In terms of prediction accuracy, our approach is ranked as being the last one out of these 5, so there are a lot to improve in order to increase its prediction accuracy. We are looking forward to find out which preprocessing step should be replaced or improved and we plan on releasing a future, more in-depth publication of the current paper, that will provide better results.

Table 5. Comparison in terms on F1 score.

Model	**F1-score**
Multinomial NB	81.05%
SCDV-MS	82.71%
LSTM-reg	87.00%
VLAWE	89.30%

Table 5 shows how the current approach compares with others in terms of F1 score. Our approach is also ranked as being the last one out of these 5, but this time the differences are not that huge, which is good to see. We're also looking forward to improve the F1 score of our algorithm and we hope that our future release will provide better results on this metric aswell.

Our next challenge is to improve the current model as much as possible and to implement new feature selection techniques. After we will be done improving the current model, we plan on migrating the application to new datasets like Reuters-21578 or similar Corpuses published by others. Finally, we plan on developing a text mining and classification tool which will be parameterized so the end-user can test different configurations and choose which one suits him the most.

5 References

Acknoowledgement: This work was supervised by Professor Daniel I. Morariu, from „Lucian Blaga" University of Sibiu, Engineering Faculty, Computer Science and Electrical and Electronics Engineering Department.

[1] Vít Novotný, Eniafe Festus Ayetiran, Michal Štefánik, Petr Sojka. *Text classification with word embedding regularization and soft similarity measure*, Brno, Czech Republic, 2020.

[2] Giannis Nikolentzos, Antoine J.-P. Tixier, Michalis Vazirgiannis, *Message Passing Attention Networks for Document Understanding*, Athens, Greece, 2019.
[3] Dr. S. Vijayarani , Ms. J. Ilamathi , Ms. Nithya, *Preprocessing Techniques for Text Mining - An Overview*, Tamilnadu, India, 2015.
[4] Daniel I. Morariu, Radu G. Crețulescu, Macarie Breazu, *Feature Selection in Document Classification*, Sibiu, Romania, 2013.
[5] Sofie Van Landeghem, Thomas Abeel, Yvan Saeys, Yves Van de Peer, *Discriminative and informative features for biomolecular text mining with ensemble feature selection*, Gent, Belgium, 2010.
[6] https://numpy.org/doc/stable/numpy-user.pdf
[7] https://scikit-learn.org/stable/getting_started.html

Ioan-Bogdan Canciu
„Lucian Blaga” University of Sibiu
Computer Science and Electrical and Electronics Engineering Department
Bulevardul Victoriei 10, 550024, Sibiu
Romania
Email: bogdan.canciu@ulbsibiu.ro