Analysis of Classification Algorithms for using in Vertical Retrieval Systems

Nemanja Popovic, Suzana Stojkovic
University of Niš
Aleksandra Medvedeva 14, 1800 Niš
{nemanja.popovic@outlook.com} {suzana.stojkovic@elfak.ni.ac.rs}

ABSTRACT: *Classification is the most solved and the most used machine learning problem. In a last few decades many classification algorithms have been developed. Because of that, when classification is needed in some problem solving, the best algorithm should always be selected. The problem that is analysed in this paper is choosing classification algorithms that can be used in vertical retrieval system for both document and query classification. We compared SVM, Multinomial Naïve Bayes algorithm, Bernoulli Naïve Bayes algorithm and Random forest. The experiments presented in the paper, show that in the long documents classification SVM and Multinomial Naïve Bayes algorithms have a similar precision (SVM is a little better), but the Multinomial Naive Bayes algorithm correctly classified 93.14% of queries, while SVM only 22.55%.*

Keywords: Information Retrieval, Vertical Retrieval, Text Classification, Naïve Bayes Classifier, SVM Classifier, Random Forest

Received: 21 August 2020, Revised 19 November 2020, Accepted 28 November 2020

DOI: 10.6025/jcl/2021/12/1/1-8

1. Introduction

Information retrieval systems (IRS) [1] are the systems the goal of which is to find the documents in the large corpus of the documents that contain information that the user needs. The corpuses that are retrieved increase very fast. For example, in the Web searching, the retrieving corpus consists of all documents on the Web. That is why finding the information that the user needs gets harder and harder. There are three basic requirements that IRS should satisfy:

• Response time should be as short as possible.

• The number of selected documents should not be too large.

• Retrieved documents should be relevant to the user query.

To improve all these parameters, vertical search (search on the given domain) [2] is often used. In the vertical retrieval sys-

tems, the documents from the corpus are classified in the set of the domains and the user specifies the query and the domain in which the search should be done. Users of the retrieval systems often are not experts in the domain of its queries, and they cannot specify the domain in the right way. Our idea is to automatically detect the query domain, i.e., to classify the queries on the similar ways as the documents in the corpus. The first step in that process is to choose the classification algorithm that can be applied in both document and query classification.

Classification problem is a very often solved machine learning problem. Its goal is to predict the value of the unknown class attribute based on the set of values of the known attributes. Now, classification is used in many science areas: in medicine (to classify the results of various analyses) in speech recognition, in OCR systems, etc. and, of course, in text classification. Text classification, except for vertical retrieval, is used in spam detecting and in sentiment analysis. Many classification algorithms have been developed and different algorithms are used in different areas: Naïve Bayes algorithm, SVM algorithm, Decision tree, Neural networks, Rule-based algorithm, K-nearest neighbours, Logistic regression, etc. More about classification algorithms can be seen in [3], [4] and [5] and in references therein.

Always when the classification should be applied, the first question is: Which algorithm should be used? Many papers compare the performance of various classification algorithms in different areas and analyse their use for document categorization (see for example [6], [7]). Query classification is more difficult, because a query is a very short text and because many algorithms that perform best in document classification are not applicable in the query classification. That is why many algorithms specialised only for query classification have been developed (see for example [8]-[10] and references therein). Many of these algorithms are applicable on specific domain, many of them are based on feedback of the user. This paper tests standard text classification algorithms with the goal of finding an algorithm applicable both to document and query classifications.

The paper is organised as follows: In Section 2, short description of the IRS is given. Section 3 presents the classification algorithms that are usually used in text classification. Section 4 analyses the performance of different classification methods in classification of big documents and short queries. Section 5 summarises the results of provided experiments and gives some possible directions for the future work.

2. IRS Systems

Information retrieval system accepts the user queries in the text form, retrieves the corpus of the natural language text documents and returns the list of ranked retrieved documents. To speed up the searching process, IRS creates internal representation of the documents known as inverted index. Inverted index contains data about all terms in the corpus (in which documents the term is appearing, how many times, etc.), i.e., the inverted index is a structure representation of the unstructured corpus. The search for relevant documents is performed in the inverted index, instead in the unstructured large corpus. Simplified scheme of the IRS is given in the figure 1.

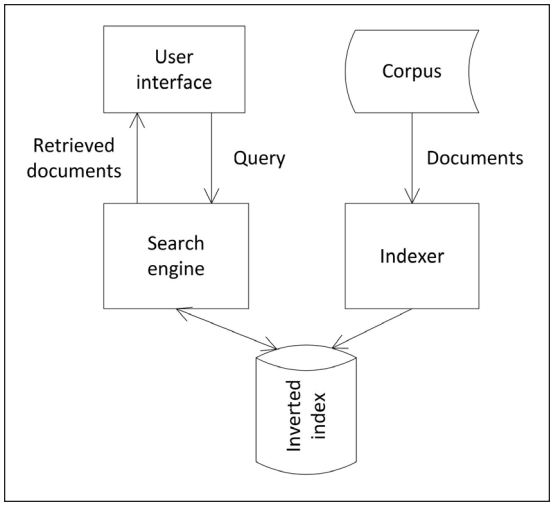

Figure 1. IRS structure

In the structure from the Figure 1, the inverted index is unique for the whole corpus. In the vertical retrieval systems, the documents are classified into domains, and separate inverted index is created for each domain. Later, searching is done only in one domain. This way the search time is reduced. If the query domain is defined correctly, the returned documents are the most relevant.

3. Classification Algorithms

In the previous section, we mentioned searching in only one domain. This type of IRS system is called IRS system with vertical search and it requires that the classification step should be performed before the search. In this section, we will discuss preparation of data set and different classification algorithms in detail.

Before running algorithms on data set, data set needs to be processed and prepared. The first step in text categorization is transforming documents, which typically are strings of characters, into a representation suitable for the learning algorithm. Information Retrieval research suggests that word stems work well as representation units and that their ordering in a document has minor importance for most classification tasks. This leads to an attribute-value representation of text. Each distinct word corresponds to a feature, with the number of times the word occurs in the document as its value. To avoid large unnecessary feature vectors, words are considered as features only if they are not "stop-words" (like "the", "or", etc.) and if they occur in the training data at least N times, where N can be configured. In addition, from Information Retrieval it is known that scaling the dimensions of the feature vector with their inverse document frequency (IDF) improves performance.

Finally, all the documents from the prepared data set can be classified using algorithms that are recommended for text classification. Those algorithms are Naïve Bayes, Support Vector Machines (SVM) and Random forest.

3.1. Naïve Bayes

Naïve Bayes algorithm is a classification technique based on Bayes Theorem. It assumes that there is no relation between the presence of different features in a class. One example of Naïve Bayes classification is fruit classification. If an object has features such as yellow, long and sweet, we can consider it a banana.

Building a model with Naïve Bayes algorithm is very easy, and it makes it very useful to work with large data sets. Even though Naïve Bayes is very simple, it is known as a very good performing algorithm and can outperform even highly sophisticated classification methods.

There are three types of Naïve Bayes algorithm and we will explain them in detail. They are Gaussian Naïve Bayes, Multinomial Naïve Bayes and Bernoulli Naïve Bayes.

Gaussian Naïve Bayes algorithm is variation of Naïve Bayes algorithm that is specifically used when the features have continuous values. It is assumed that all the features are following a Gaussian distribution. Gaussian distribution is a normal distribution.

Multinomial Naïve Bayes is a Naïve Bayes variation that estimates the conditional probability of a particular word/term/token given a class as the relative frequency of the term t in documents belonging to class c.

$$P(t \mid c) = \frac{T_{ct}}{\sum_{te V} T_{ct}}$$

This variation takes into account the number of occurrences of term t in training documents from class c, including multiple occurrences.

Bernoulli Naïve Bayes is a Naïve Bayes variation that generates a Boolean indicator about each term of vocabulary. If a term does not belong to the vocabulary then 0 is generated, if a term belongs then 1 is generated. This variation generates a significantly different model from Multinomial because it does not take into account the number of occurrences of each word and because it takes into account the non-occurring terms within documents. Bernoulli model is known to make many mistakes while classifying long documents because it does not take into account the multiple occurrences of the words.

3.2. SVM

Support vector machines are based on the Structural Risk Minimization principle ([11], [12]) from the computational learning theory. The idea of structural risk minimization is to find a hypothesis *h* for which we can guarantee the lowest true error. The true error of *h* is the probability that *h* will make an error on an unseen and randomly selected test example. An upper limit can be used to connect the true error of a hypothesis *h* with the error of *h* on the training set and the complexity of *H* (measured by VC-Dimension), the hypothesis space containing *h* [9]. Support vector machines and the hypothesis *h* which (approximately) minimises this limit on the true error by effectively and efficiently controlling the VC-Dimension of *H*.

SVMs are universal learners. In their basic form, SVMs learn linear threshold function. Nevertheless, by a simple "plug-in" of an appropriate kernel function, they can be used to learn polynomial classifiers, radial basic function (RBF) networks and three-layer sigmoid neural nets.

One remarkable property of SVMs is that their ability to learn can be independent of the dimensionality of the feature space. SVMs measure the complexity of hypotheses based on the margin with which they separate the data, not the number of features. This means that we can generalise even in the presence of very many features, if our data is separable with a wide margin from the hypothesis space.

The same margin argument also suggests a heuristic for selecting good parameter settings for the learner (like the kernel width in an RBF network) [9]. The best parameter setting is the one that produces the hypothesis with the lowest VC-Dimension. This allows fully automatic parameter tuning without expensive cross-validation.

3.3. Random Forest

Decision tree learning is a method for approximating discrete-valued target functions, in which a decision tree represents the learned function. Learned trees can also be represented as sets of if-then rules to improve human readability. These learning methods are among the most popular of inductive inference algorithms and have been successfully applied to a broad range of tasks from learning to diagnose medical causes to learning to assess credit risk of loan applicants.

Random forest is a method (firstly defined in [13]) that use sets of decision trees on either random subsets of training data, or splits with randomly generated vectors, and computes the score as a function of these different components. Usually these random vectors are generated from a fixed probability distribution. Because of this, random vectors can be created by either random input selection, or random split selection. In addition, it is possible to create the trees in a lazy way, which is tailored to the particular test instance at hand in the case of random forests.

4. Documents and Query Classification by Different Algorithms

It appears there is a new problem in testing the system we used: It needs to have the corpus of classified documents and corpus of classified queries and for each pair (<document>,<query>) it should assess the relevance of the document to the query. There are many benchmark corpuses for document classification testing, and some corpuses of documents and queries for testing of information retrieval systems (such as TREC [12]), but there are not known corpuses that can be used in both purposes. Because of that, we created our own corpus containing 1225 documents that are taken from the Wikipedia website, and 100 queries suitable for searching in the document corpus. The documents and queries are classified into 10 classes: architecture, art, biology, chemistry, computer science and informatics, literature, mathematics, music, philosophy and physics.

To verify that the classification method is applicable on bigger corpuses, we tested all presented algorithms on the standard benchmark corpus of documents Reuters 21578- Apte-90Cat [14]. This corpus contains 15473 documents classified into 91 different classes.

Implementation of classifiers is done using Java programming language. All classification algorithms are used from Weka 3.6.6 library. Classification model is created by using StringToWordVector filter from Weka. This filter is used to get 1000 words from each document. IDFTransform and TDTransform are turned on and stop list was used to filter out all stop words from documents. After this filter is applied, classifications are performed on the transformed corpus.

We conducted the experiments on a HP ZBook 15, the basic parameters of which are shown in Table 1.

CPU	Intel® Core™ i7-4900MQ CPU @ 2.80GHz
RAM	16GB
OS	Windows 8.1 Enterprise
GPU	NVIDIA Quadro K2100M

Table 1. Experimental System Performances

The results of the document classification from the Wikipedia corpus are shown in the Table 2. This table contains number and percentage of correctly classified instances and time needed for model creation. All values are average values calculated after running algorithms 10 times.

	Number/percentage correctly classified	**Time for model creation (s)**
Random Forest	1141 / 93.2%	9.05s
Naïve Bayes Multinomial	1178 / 96.1%	0.04s
Bernoulli Naïve Bayes	1120 / 91.4%	2.45s
SVM	1161 / 94.8%	2.4s

Table 2. Classification Performances on Wikipedia Corpus

As it can be seen from Table 2, the best results are achieved by using Naïve Bayes Multinomial algorithm. The number/percentage of correctly classified documents with Naïve Bayes Multinomial algorithm is 1178/96.1% that is better than SVM algorithm is used (1164/94.8%). The time required for model creation is also showing that best choice is Naïve Bayes Multinomial, as the time of 0.04s is much shorter than SVM with 2.4s.

	Number/percentage correctly classified	**Time for model creation (s)**
Random Forest	2502 / 62.1769%	2716.38
Naïve Bayes Multinomial Bernoulli+	2867 / 71.25%	3.27
Naïve Bayes	2647 / 65.78%	56.32
SVM	2923 / 72.64%	87.57

Table 3. Classification Performances on Reuters 21578 Corpus

All algorithms are applied on classification of documents from the "Reuters 21578-Apte-90Cat" corpus. Results of these experiments are shown in Table 3.

From Table 3 we can see that SVM has slightly better results than Naïve Bayes Multinomial, but it is around 26 times slower. Because SVM and Naïve Bayes Multinomial have similar number of correctly classified documents, but Naïve Bayes Multinomial algorithm is much faster, this makes Naïve Bayes Multinomial algorithm a better choice for vertical IR system.

Results of executing classification algorithms on the search query corpus are given in Table 4. The results from Table 4 show that the most applicable algorithm in query classification is Naïve Bayes Multinomial algorithm.

	Number/percentage correctly classified
Random Forest	12/11.77%
Naïve Bayes Multinomial	95/93.14%
Bernoulli Naïve Bayes	18/17.65%
SVM	23/22.55%

Table 4. Classification Performances on Search Query Corpus

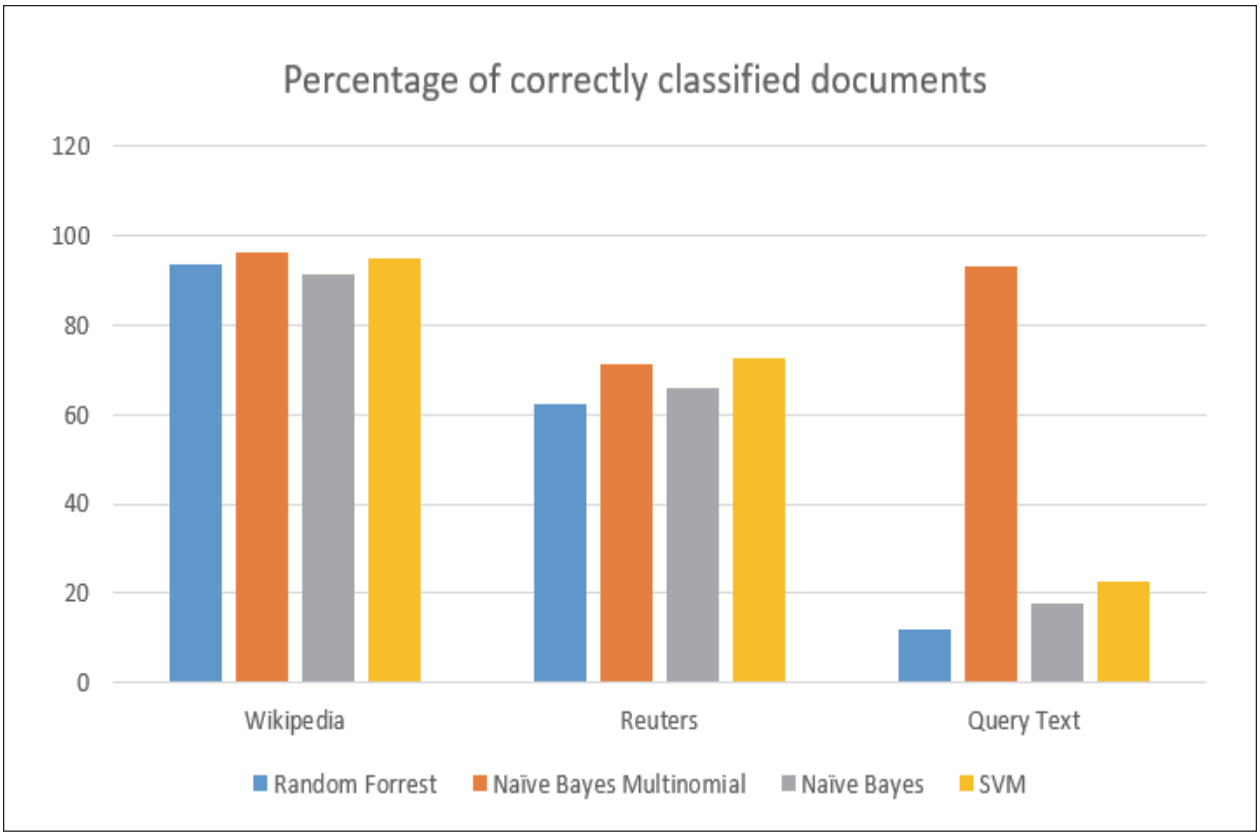

Figure 2. Comparison of classification algorithms on different testing corpuses RS structure

A comparative review of the correctly classified documents by all algorithms on all three corpuses is shown in the Figure 2.

5. Conclusion

After looking into the results of Random Forest, Naïve Bayes Multinomial, Bernoulli Naïve Bayes and SVM that were compared on two different corpuses of documents, we can clearly see that Naïve Bayes Multinomial and SVM algorithms which have the best results in terms of the number of correctly classified documents. Naïve Bayes Multinomial shows a better time for model creation and for running classification. However, based on the best results in the query classification, the Naïve Bayes Multinomial is the best solution for the vertical retrieval systems.

For future work, the system can be improved by using different classification methods for document and query classification. In that case, for query classification some specialised algorithm for query classification or for short text classification (such as LSA [16], Bag of concepts [17] etc.) can be applied.

References

[1] Manning, C.D., Raghavan, P., Schutze, H. (2009) *An Introduction to Information Retrieval*, Cambridge University Press, Cam bridge, England, 2009.

[2] John, B. (2005). *The Search: How Google and its Rivals Rewrote the Rules of Business and Transformed Our Culture*. New York: Portfolio, 2005.

[3] Duda, R.O., Hart, P. E., Stork, D. G. (2000). *Pattern Classification*, Second edition, Wieley Interscience, 2000.

[4] Han, J., Kamber, M. (2006). *Data Mining: Concepts and Techniques*, Second edition, Elsevier, 2006.

[5] Tan, P.-N., Steinbach, M., Kumar, V. (2006). *Introduction to Data Mining*, Addison Wesley, 2006.

[6] Mahinovs, A., Tiwari, A. (2005). Text Classification Method Review, *Decision Engineering Report Series,* Edited by Rajkumar Roy and David Baxter, Cranfield University, 2005.

[7] Li, Y. H., Jain, K. (1998). Classification of Text Documents, *The Computer Journal*, 4(18) 537-546.

[8] Le, D.-T., Bernardi, R. (2012). Query Classification Using Topic Models and Support Vector Machine, *2012 Student Research Workshop*, p19-24, Jeju, Republic of Korea, 2012.

[9] Alemzadeh, M., Karray, F., Khoury, R. (2012). Query Classification using Wikipedia's Category Graph, *Journal of Emerging Technologies in Web Intelligence*, 4 (3), p 207-220, 2012.

[10] Xia, C., Wang, X. (2015). Graph-Based Web Query Classification, 12th *Web Information System and Application Conference*, p 241-244, 2015.

[11] Cortes, C., Vapnik, V. (1995). Support-Vector Networks, *Machine Learning*, vol. 20, p. 273-297.

[12] Vapnik, V.N. (1995). *The Nature of Statistical Learning Theory*, Springer, New York.

[13] Ho, T.K. (1995). Random Decision Forest, 3rd *International Conference on Document Analysis and Recognition*, Montreal, QC, p 278-282,1995.

[14] Reuters-21578 Text Categorization Collection available at http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html (last access 14.3.2017)

[15] TREC Retrieval Conference, available on http://trec.nist.gov/ (last access 14.3.2017)

[16] Dumais, S.T. (2005). Latent Semantic Analysis, *Annual Review of Information Science and Technology*, 2005.

[17] Sahlgren, M., Cöster. R. (2004). Using Bag-of-Concepts to Improve the Performance of Support Vector Machines in Text Categorization, *20th Intern. Conf. on the Computational Linguistics*, Article no. 487, Geneva, Switzerland, 2004.