

An OKAPI-Based Approach for Article Filtering

Benedict Lee and Cuong Van Than
Department of Computer Science
Rice University
{benlee, cvthan}@rice.edu

1 Introduction

Text classification is the task of assigning predefined categories to documents. This problem has become important, both because of the vast volume of online data produced these days, and of its various practical applications. Typical applications include organizing Web pages by topic, filtering out inappropriate online content for specific audiences, and detecting authorship of disputed works.

Text classification can also be used as a pre-processing step in other machine learning tasks. For example, the Ares Project [Devika] aims to predict international conflict based on events extracted from news articles from sources such as the AFP, Reuters, and the Associated Press. The proliferation of news in electronic form has made it possible to automate the forecasting of conflict outbreaks. However, the huge number of articles also imposes a challenge to the accuracy of the prediction process because news sources also contain articles that are irrelevant to political tensions. When building an event database, we might get, for example, sports or music-related stories, which have nothing to do with international conflict, so we do not want to include such articles in our database. The main goal of our project for Comp 540 is to develop a process that differentiates politically relevant articles from politically irrelevant articles.

2 Overview of Text Classification

In this section, we provide a brief overview for the text classification problem. The book “Learning to Classify Text Using Support Vector Machines” by Joachims gives an in-depth discussion for what is being presented here.

2.1 Definition

The general problem of text classification can be formally stated as learning an optimal estimator, based on training examples, to the function $\varphi : D \times C \rightarrow \{True, False\}$, where D is the document domain and C is the set of predefined classes. We often assume that no additional knowledge or metadata is available, so documents are classified solely on their content. Depending on the set C and on different tasks, classification may be binary, multi-class or multi-label.

In the binary setting, each document is classified in exactly one of two categories. The two categories might be “relevant” and “irrelevant”, as in our article filter, “like” vs. “dislike”, or “spam” and “non-spam” in de-spam email programs. Though binary classification is the simplest task, it is the most important in the learning problem [Joachims].

In case C has more than two classes, we have the multi-class classification problem. A typical application is to assign books into different subjects such as “mathematics”, “physical sciences”, “engineering”, or “so-

cial sciences” by the book titles and short descriptions (e.g. editorial reviews as in Amazon.com).

We can always solve a multi-class classification task by a number of binary classifiers. The common strategy is *one-against-the-rest*. Label the first element of C as +1 and all the remaining elements as -1 and apply a binary algorithm to classify documents according to these two labels. For the class -1, we repeat to split into two labels as above until every label corresponds to exactly one element of C .

Another method for dealing with multi-class problem by using binary classifiers is the *pair-wise* strategy: We learn each classifier for every pair of classes of C (so we have $O(|C|^2)$ classifiers), and an unseen document is classified by majority vote of these classifiers.

In practice, a document can fall into more than one class. For example, the book “Elements of Statistical Learning” used in our course falls into both statistics and computer science subjects. When a document can be assigned to more than one class as in that example, we have the multi-label text classification problem. Currently, no algorithms directly handle it. However, one can still use binary classifiers to learn a model for it. The prediction for a document in this setting is now an l -dimensional binary vector in $\{-1, +1\}^l$, where l is the total number of class. Under this representation, the multi-label problem gets solved by independently applying a binary classification algorithm for each category.

So a good binary classifier is of critical importance. Our project aims to develop a text classification program that has a better performance than that of the current naive Bayes classifier used in Ares.

2.2 Text Representation

The next step is to choose an appropriate representation for texts, which has a crucial influence on the performance of classification algorithms. In fact, the choice of a representation of articles also affects the choice of an algorithm and the design of the whole program. There are typically five types of text representation [Joachims]:

- Sub-word level—decomposition of words. In n -grams, the most popular sub-word representation, the document is represented as a set of strings of n characters. The word “book” corresponds to “boo” and “ook” in 3-grams. See [Neumann].
- Word level—words and lexical information. The text is split into a sequence of words. It is usually assumed that the order of words is independent, so only the frequencies of words are recorded, while the overall structure of the document is ignored.
- Multi-word level. At word level representation, it would not be possible to distinguish between a verb and noun “book,” for example. Basic building blocks in multi-word level are generally phrases and sentences so that it can capture syntactic information.
- Semantic level. This would be the most efficient level of representation because it captures the meaning of the whole text.
- Pragmatic level—the meaning of the text with respect to context and situation. This level is often not applicable to online news articles, as such context information is barely available.

Our algorithm chooses to represent documents at the semantic level by computing two values—the relevance and irrelevance rank. Because these scores measure the similarity of the whole article against a training dataset, we expect our program to perform better than programs that choose the representation at lower levels.

2.3 Feature Selection

Before we learn a model for a text classification task, the training dataset, after being converted to an appropriate representation, needs refinement. The main purpose of this additional step is to remove irrelevant attributes, such as prepositions and conjunctions, from the representation. Another motivation is computational efficiency—eliminating unnecessary and very common words like “the” or “and” effectively reduces the dimension of the input space.

There are two kinds of feature selection [Joachims]:

- Feature subset selection. This technique retains a subset of the original attributes. As mentioned in the previous paragraph, stopwords like “the” or “and” are irrelevant and they should be removed from the feature vector. Yang and Pedersen [Yang97] also suggest to delete infrequent words from the text representation, based on an assumption that words rarely occur through the training dataset have negligible influence on classifying texts.
- Feature construction. New features are introduced by combining original attributes. English words often have several variations. For example, words “defense” “defend” and “defensive” have the same stem. It would be more appropriate to consider different word forms of a stem altogether as a single attribute.

The net effect is minimizing the number of attributes. Another feature construction technique is to group synonyms into equivalence classes. This results in a more robust text classifier, because different words having similar semantic meanings are treated equally and as a single feature. [Fisher94] explores the use of this approach in text classification.

2.4 Conventional Learning Methods

We do not aim to provide an exhaustive list and description of learning methods used in classification. Rather, we list here some of the most popular approaches. However, we will discuss the naive Bayes classifier at a greater depth in “Related problems” Section, as it is the current method employed by Ares.

- Naive Bayes classifier. The Bayesian approach is to assign a document d to a class y that maximizes the probability $\Pr(y | d)$. From Bayes theorem, this posterior probability is computed by:

$$\Pr(y | d) = \frac{\Pr(d | y) \Pr(y)}{\Pr(d)}.$$

Because $\Pr(d)$ is independent of y , maximizing $\Pr(y | d)$ is equivalent to maximizing $\Pr(d | y) \Pr(y)$. $\Pr(y)$ can be estimated as the frequency of documents in class y in the training dataset. With the assumption that attributes of d are independent, the likelihood $\Pr(d | y)$ can also be estimated in the same way as $\Pr(y)$.

- k -nearest neighbors. The method classify a new document by computing the similarity between it and other documents, and then assigning it the majority class among k closest neighbors. k -nearest neighbor algorithms show a good performance on text categorization tasks

[Yang97]. In this method, an appropriate similarity/distance metric is essential to a good classifier. Our text classification program is in this learning category and it makes use of the OKAPI similarity measure. We will discuss more about this metric in Section [OKAPI].

- Decision trees. The most popular decision tree learning program is C4.5 by Quinlan [Quinlan93]. The approach is to pick up the best attribute, create tree nodes corresponding to possible values of this attribute, and divide training examples among these nodes. The process is then repeated until no examples are left. To decide which attribute to be used at each step, information gain is calculated for each attribute (based on the training dataset). The attribute with the highest gain is chosen to as the best classifier for this step. Informally, information gain measures how well an attribute separates the training set. Therefore, the higher the gain, the better separation resulting from classifying training examples on the associated attribute. For a formal concept, equations for computing information gain and an illustrative example see [Mitchell].

3 Related Work

Text classification is a long-established problem, so there has been much related work in this field. In this section, we look at some applications of text classification as well as the current naive Bayes classifier used Ares.

3.1 Ares and the Naive Bayes Classifier

Classification of news articles is an integral part of the Ares Project. The first stage in achieving the ultimate goal of predicting international conflicts is to build an event

database. The current version of Ares uses a naive Bayes classifier to classify articles. Figure 1 illustrates how the naive Bayes classifier is used to build an event database.

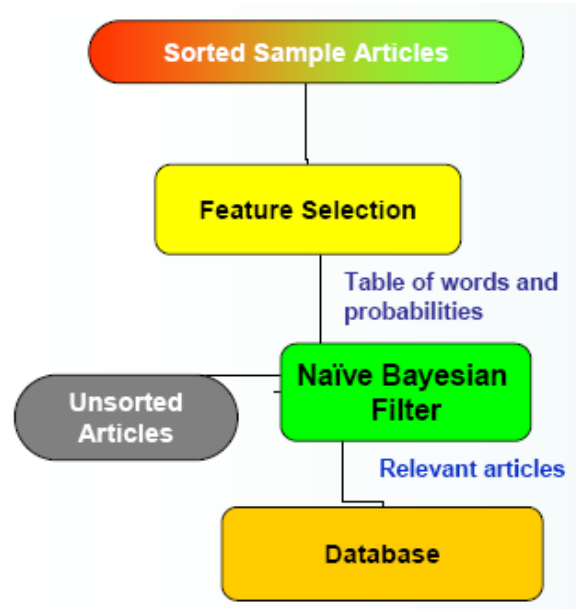


Figure 1: Naive Bayes classifier in Ares. [From Ares' Website]

As mentioned in Subsection 2.4, a document d is assigned to class y if the probability $\Pr(y | d)$ is maximized, which is equivalent to maximizing $\Pr(d | y) \Pr(y)$. To estimate $\Pr(d | y)$, we assume that words occurring in the document are independent and identically distributed. Even with this assumption, however, a rough calculation [Mitchell] shows that the total number of probabilities $\Pr(d_i | y)$ that must be estimated is about $2 \times 50,000$. To make that number more manageable, the naive Bayes classifier in Ares decided to only select the 2000 most relevant words. This strategy effectively reduces the number of terms $\Pr(d_i | y)$ by 25 times. However, it also makes the classifier sensitive to the selection of feature words. See Ares' Website for experimental results of this naive Bayes classifier.

3.2 Support Vector Machines

A support vector machine (SVM) is a class of supervised machine learning techniques. It is based on the principle of structural risk minimization. In linear classification, support vector machines create a hyper plane that separates the data into two sets with the maximum-margin. (A hyper plane with the maximum-margin has the distances from the hyper plane to points in the two sides are equal.) Mathematically, SVMs learn the sign function $f(x) = \text{sign}(wx + b)$, where w is a weighted vector in \mathbb{R}^n . SVMs find the hyper plane $y = wx + b$ separating the space \mathbb{R}^n into two half-spaces with the maximum-margin. Linear SVMs can be generalized for non-linear problems. To do so, the data is mapped into another space H and we perform the linear SVM algorithm over this new space.

3.3 Applications of Text Classification

There are a number of text classification applications. A well-known application is spam filtering; e-mail needs to be classified as either legitimate or unsolicited/unwanted (spam). In [Graham], Graham proposes the use of naive Bayesian filtering in the context of spam classification. While Bayesian filtering has proven to be fairly successful in this problem, it can also be used as a more general text classifier. Following is a short list of other applications [Fabrizio]:

- Detecting authorship of disputed works. Text classification can be used for the purpose of detecting authorship of disputed works. For example, there are many works that are known to belong to Shakespeare, but there are other stories whose authorship is in doubt.
- Text filtering. The Web is flourished with online articles and stories. Many of

them contain adult contents, and we want to shield children from reading them. So, we can integrate a text classifier into Web browsers to filter out not-for-kids stories.

- Filing classified ads into classes. For example, should we print an ads under the category “Car Rental,” or “Real Estate.”
- Filing patents under predefined categories.

4 Approach

We believe that relevant articles are more similar to other relevant articles than irrelevant ones. With this idea in mind, we propose a method to classify articles based on “similarity.” Given a new article, we generate a relevance rating and an irrelevance rating based on how “similar” the given article is to the training set of pre-classified articles.

While there are many ways to define “similarity”, we choose to use the OKAPI BM25 ranking formula [10]. OKAPI is a well-known and widely used ranking formula. Furthermore, Microsoft SQL Server 2005, the database used in the Ares project, provides built-in support for OKAPI. For future work, we could see how well other ranking formulas performed.

Given two bodies of text, the OKAPI formula will generate a score that corresponds to the similarity of the two bodies of text; higher scores indicate better matches.

For the problem of classification, we first need to have a sufficiently large training set of pre-classified articles. It needs to be large in order to have the breadth necessary to make matches with any input article. These articles are separated into “Relevant” and “Irrelevant” training sets. To calculate a relevance rating, we use the OKAPI formula

to get scores comparing an input article with every article in the “Relevant” training set. We only take the sum of the top N scores and set that as the relevance rating for the inputted article. Similarly, we use OKAPI to get similarity scores between the inputted article and every article in the “Irrelevant” training set and set the irrelevance rating as the sum of the top N scores.

Once we’ve obtained a relevance and irrelevance rating for a new article, we simply determine classification based on which score is higher; if the relevance rating is higher than the irrelevance rating for a given article, we will classify the article as relevant, and vice versa. While the naïve Bayesian approach gives individual weights to every word, this approach will compare articles as a whole with each other. We believe that this will be a more confident indicator about an article’s content, which is especially important for classifying it as in *Relevant* or *Irrelevant* category.

4.1 Top N Selection

The selection of the top N scores serves as a form of artificial clustering. Even within the “Relevant” training set, we expect that there will be a lot of articles that simply do not match well with an inputted relevant article. Thus, we only want to see the best matches, as we believe that these articles are the ones that should be the determining factor in classification. In our tests, we plan to see if changing N has an impact on performance.

5 Experiments

5.1 Evaluation Metrics

To evaluate the performance of our approach, as well as any other text classification task, we use the following metrics.

- Recall - Defined as the probability that a relevant article is correctly classified. The recall measure is estimated by $n_{11}/(n_{11} + n_{01})$, where n_{11} and n_{01} are the number of relevant articles correctly and incorrectly classified, as shown in Figure 2.
- Precision - Defined as the probability that an article classified as relevant is, in fact, relevant. A simple estimate for precision is $n_{11}/(n_{11} + n_{10})$.
- Accuracy - Defined as the probability that an article, both relevant and irrelevant, is correctly classified. Its estimate is $(n_{11} + n_{00})/(n_{11} + n_{01} + n_{00} + n_{10})$.

	label <i>Relevant</i>	label <i>Irrelevant</i>
prediction <i>Relevant</i>	n_{11}	n_{10}
prediction <i>Irrelevant</i>	n_{01}	n_{00}

Figure 2 Contingency table for binary data.

5.2 Reuters Experiments

For our first set of experiments, we chose to work with a dataset containing ~180,000 categorized Reuters articles from September 1996 through July 1997. In order to use these articles for the training set, we used Reuters’ pre-assigned categories to determine relevance.

Relevant categories:

- GVIO – War, Civil War
- G13 – International Relations
- GDIP – International Relations

Irrelevant categories:

- 1POL – Current News – Politics
- 2ECO – Current News – Economics
- 3SPO – Current News – Sports
- ECAT – Economics

- G12 – Internal Politics
- G131 – Defense
- GDEF – Defense
- GPOL – Domestic Politics

Any Reuters articles that did not fall in these categories were thrown out. In order to make a fair comparison, we used the exact same training sets among the different algorithms.

Using the Reuters dataset, we performed 10-fold cross-validation on various filtering algorithms to see how well they performed using the aforementioned metrics. The approaches that we compared were Naive Bayes (NB), Weight-based complement Naive Bayes (WNB), and our OKAPI-based approach.

In 10-fold cross-validation, we randomize the order of the articles and divide the set up into 10 chunks. For each chunk, we use that chunk as the testing set and the remaining 9 chunks for the training set.

In our OKAPI-based approach, we chose N values of 5, 10, 25, and 50, to see if the selection of top N scores significantly affects the performance of the OKAPI-based approach. Furthermore, we also made a composite filter that takes in the classifications from each of the 4 N values and determines a classification based on majority rule, leaning in favor of *Relevance* in the case of ties.

Classifier	Recall	Precision	Accuracy
NB	85.86%	80.64%	89.47%
WNB	86.62%	79.78%	89.30%
OKAPI(5)	92.64%	86.83%	91.51%
OKAPI(10)	93.09%	87.26%	93.79%
OKAPI(25)	93.12%	87.27%	93.80%
OKAPI(50)	93.19%	86.91%	93.68%
OKAPI (C)	94.15%	86.18%	93.65%

Figure 3 Reuters results.

The results of these experiments are shown in Figure 3. The first observation we make is that the OKAPI-based approach performed better than NB and WNB across the board. We believe that this is because only the most important articles make the decision in the OKAPI-based approach, whereas each article in the training set has a voice in the NB and WNB approaches. If some form of clustering were incorporated with the NB and WNB approaches, we believe that their performance will improve.

Within the OKAPI-based approaches, There isn't much difference between the various N values, save for N = 5, which has the worst results for the OKAPI-based filters. We believe this is because the top 5 OKAPI scores are not sufficient to make as conclusive judgments as the other OKAPI-based approaches can. For N = 10, 25, and 50, the percentages are fractions of a percentage point away from each other. This shows that obtaining even more scores does not necessarily increase performance, as long as a sufficient number of top few are selected. In fact, the accuracy decreased as N increased from 25 to 50, which shows that having even larger N could potentially hurt performance.

Finally, we observe that the composite OKAPI filter has the highest recall among all approaches. Due to the voting scheme, the composite approach favors *Relevant*, reducing the number of false irrelevant classifications but increasing the number of false relevant classifications. Even with this tradeoff, the accuracy remains on par with the other OKAPI-based approaches. In the scope of the Ares project, we are more willing to allow irrelevant articles in the pool in order to reduce the number of relevant articles that are thrown away. Taking this stance, the composite approach is perhaps

the most suited of the OKAPI-based approaches to tackle this problem.

5.3 AP/BBC Tests

In our second set of experiments, we use the Reuters dataset as the training set and see how well the approaches perform on a dataset from other sources.

Thanks to help from Professor Richard Stoll of the Political Science department at Rice University, we obtained relevant/irrelevant classifications of 1000 AP and BBC articles from a similar date range.

Classifier	Recall	Precision	Accuracy
NB	53.53%	40.95%	56.10%
OKAPI(5)	86.76%	42.25%	53.20%
OKAPI(10)	86.20%	41.80%	53.50%
OKAPI(25)	87.61%	42.03%	52.70%
OKAPI(50)	89.30%	42.55%	53.40%
OKAPI (C)	90.14%	41.45%	51.30%

Figure 4 AP/BBC results.

The results of our tests are shown in Figure 4. The Naïve Bayes and OKAPI-based filters both fared much worse on this experiment.

One reason why the filters did worse is because the actual classifications for the AP/BBC set were done in a slightly different manner than in the Reuters tests. Professor Stoll assigned relevance based on whether an article contained an encodable event, as opposed to what category the article would fall under. We hope that the two sets would largely overlap, but we have not performed the necessary studies to see if this is true.

We notice that the OKAPI-based approach has better recall and precision than naïve Bayes, but it has worse accuracy. This is because the OKAPI-based approaches make

significantly more false relevant classifications than naïve Bayes did. For this testing set it certainly seems like the OKAPI-based approach leans more towards classifying an article as relevant over irrelevant.

Our initial thinking of why the OKAPI-based approach does poorly on the irrelevant articles is because the training set simply does not have enough breadth. However, when we augmented the training set with the entire Reuters dataset, our performance did not improve, as is shown in Figure 5.

(Add Figure 5 here)

If we were to only add more irrelevant articles to the training set, we would expect that the performance on classifying irrelevant articles is true. However, we see the exact opposite trend. Trying to figure out why we were observing this, we noticed that the relevant training set increased from ~45,000 articles to ~61,000, which means that all of the relevant articles may not have been originally used. The larger relevant training set could be the reason why our results improve on classifying relevant articles but degrade when classifying irrelevant ones.

(Add more stuff here)

6 Conclusion

We believe that there is the OKAPI-based approach certainly shows promise in the task of article filtering. While it performed quite well on our Reuters tests, it fared much worse on the AP/BBC tests (though so did the NB classifier).

(More stuff to be added)

7 Future Work

(Add stuff here)

8 References

- [1] P. Graham. "A Plan for Spam", [online] 2002, <http://www.paulgraham.com/spam.html>.
- [2] L.D. Baker and A.K. McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval*, pages 96-103. ACM Press, 1998.
- [3] T. Joachims. Learning to classify text using support vector machines. Kluwer Academic Publishers, 2001.
- [4] Subramanian. "Rice Event Data System", [online] 2005, <http://hyperion.cs.rice.edu>.
- [5] D. Lewis. An evaluation of phrasal and clustered representation on text categorization task. In *International ACM SIGIR conference on research and development in information retrieval*.
- [6] M.K. Buckland and F. Gey. The relationship between Recall and Precision. *Journal of the American Society for Information Science* 45, no. 1 (Jan. 1994): 12-19.
- [7] G. Neumann and S. Schmeier. Combining shallow text processing and machine learning in real world applications. *IJCAI99 Workshop on machine learning for information filtering*, 1999.
- [8] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69-90.
- [9] D.E. Fisher. Topic characterization of full length texts using direct and indirect term evidence. Technical Report CSD-94-809, University of California, Berkeley.
- [10] [OKAPI paper]
- [11] J.R. Quinlan. C4.5: Programs for Machine Learning. Machine Learning. Morgan Kaufman, San Mateo, CA.
- [12] T.M. Mitchell. Machine Learning. McGraw-Hill, 1997.
- [13] Fabarizio Sebastiani. Text Classification for Web Filtering. Final POESIA Workshop: Present and future open-source content-based Web filtering. Pisa, Italy.