Machine Learning Methods for Document Classification

(**Modified** version of the presentation)

Hao Luo (RUID: 162006265)
Yun Wang(RUID: 154000804)

1 Introduction

The ability to automatically classify documents in to a fixed set of categories is highly desirable in many present world scenarios including classifying a large amount of archival documents such as academic papers, newspaper articles, and business archives. Take newspaper for example, the articles can be classified in to various categories such as 'business', 'sports', 'technology' and so on. There are also other scenarios involving classifying documenters as they are created, including classifying movie review articles or comments into 'positive' or 'negative' reviews or classifying blog entries using a fixed set of labels.

Natural language processing offers powerful techniques for automatically classifying documents. These techniques are predicated on the hypothesis that documents in different categories distinguish themselves by features of the natural language contained in each document. Salient features for document classification may include word structure, word frequency, and natural language structure in each document.

Our project focuses on applying machine learning to automatically classify newspaper articles from Reuters Corpus. The aim of our project is to investigate and implement techniques and determine which techniques perform well to automatically article classification.

This project experiments with different natural language feature sets as well as different statistical techniques using these feature sets and compares the performance in each case. Specifically, our project involves experimenting with feature sets for Frequency Counting Based Classification, Naive Bayes Classification, and tf.idf-Support Vector Machine Classification.

The paper proceeds as follows: Section 2 discusses related work in the areas of document classification and gives an overview of each classification technique. Section 3 details our approach and implementation. Section 4 shows the results of testing our classifiers and analysis of results. In Section 5, we give the conclusion and discussed possible future of this project.

2 Related Work and Overview of Classification Techniques

There have been a variety of machine learning techniques that have demonstrated reasonable performance for document classification. Some of these techniques includes neural networks[1], support vector machines, genetic programming [2], rule learning algorithms [3], and Maximum Entropy Classification[4].

For this project, we focus on testing and comparing the performance of Frequency Counting Based Classification, Naive Bayes Classification [5, 6, 7] and tf.idf-Support Vector Machine Classification[8, 9]. And based on the original model, we propose a modified TF-IDF method for classification

2.1 Frequency Counting Based Method

Frequency Counting Based Classification is a very primary approach to classify documents. Each distinct word is a feature and just assigned with the occurrence of the word in the document as its value.

2.2 Naive Bayes Classification

This subsection cites material from [7] extensively to explain the basics of Naive Bayes Classification.

Bayesian classifiers are probabilistic methods that make strong assumptions about how the data is generated, and posit a probabilistic model that embodies these assumptions. Bayesian classifiers usually use supervised learning on training examples to estimate the parameters of the generative model. Classification on new examples is performed with Bayes' rule by selecting the category that is most likely to have generated the example.

The naive Bayes classifier is the simplest of these classifiers, in that it assumes that all features of the examples are independent of each other given the context of the category. This is the so-called "naive Bayes assumption." While this assumption is clearly false in most real-world tasks, naive Bayes often performs classification very well. Despite this, practical applications of naive Bayes classifier have had high degrees of accuracy in many cases.

The naive Bayes classifier is much simpler and much more efficient than other supervised learning. This is largely because of the independence assumption which allows the parameters for each feature to be learned separately. In the case of document classification, number of features is document classification is usually proportional to the vocabulary size of the training document set. This number can be quite large in many cases so the efficiency of the naive Bayes classifier is a major advantage over other classification techniques.

2.3 tf.idf-Support Vector Machine Classification

2.3.1 tf.idf

Tf-idf stands for term *frequency-inverse document frequency*, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query[10, 11].

2.3.2 Support Vector Machine

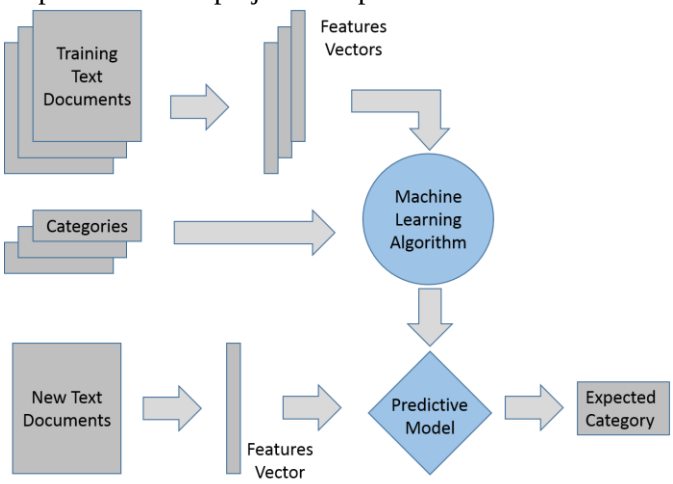
Introduced in 1992, Support Vector Machines (SVM) has become regarded as one of the most powerful learning algorithms for text categorization[8]. SVM is most commonly used to split a single input set of documents into distinct subsets. For example, we could classify documents into privileged and non-privileged sets. However, it is often necessary to group documents into more than two classes. Therefore, many methods using SVM have been established to support multi-class classification. For each category the classifiers are trained on a labeled set where the documents from the corresponding category make up positive examples and rest of the documents become negative examples. In order to classify new data, each classifier is used to produce a confidence value of a point belonging to the corresponding category and the category with greatest confidence is selected.

3 Approach

3.1 Problem Specific

Our project focused on test and comparing different classification methods for Document Classification. We implemented these methods on newspaper articles from Reuter's Corpus.

Reuter's corpus has 10860 documents from 90 different categories. These documents have been pre-split to training and test set which containing 7769 and 3091 documents, respectively. It is worthy to mention that all the documents are crossly categorized and number of documents belonging to categories varies severely. For example, category 'earn' has 2877 documents but category 'castor-oil' only 1 document. Implementation process of this project is depicted in scheme 1.



Scheme1 Implementation process of this project

3.2 Feature Selection

1. In our feature-sets we removed those words appear in stop-words list from training set. Those words, such as "according", "and", "in" and so on, occur frequently but contain very little information.
2. We employ Information Gain Method (I.G) and choose words with high mutual information. Finally, we select the top 20000 informative words in our feature-sets.

3.3 Frequency Counting Based Classification

Frequency Counting Based classifier just assign the occurrence of word in different categories as value of the feature.

When applied to a new test document, the classifier compute the prediction value of categories of the test document using the following function and return the category with highest value as target answer.

$$p(c_i|w_1, w_2, \dots, w_n) = \sum_{i=1}^{n} p(w_i|c_i)$$

where $p(w_i|c_i)$ is the frequency of word w_i in category c_i.

We use this method as a control to evaluate the performance of other methods.

3.4 Naive Bayes Classification

The probability model for a classifier is conditional model given by $p(C|F_i, \dots, F_n)$ over an independent class variable C representing one of the possible classification labels, conditioned on several feature variables F_1 through F_n. Using Bayes' theorem this can be rewritten as:

$$p(C|F_1,\dots,F_n) = \frac{p(C)p(F_1|C)p(F_2|C,F_1)\dots p(F_n|C,\ F_1,F_2,\dots,F_{n-1})}{p\,(F_1,\dots,F_n)}$$

Naive Bayes Model assumes that all features of the training documents are independent of each other given the context of the category called the "naive Bayes assumption" or "independent feature assumption." Using this assumption, the conditional probabilities for each feature F_i in the numerator of the above equation become conditional only on C. This means the above equation can be rewritten as:

$$p(C|F_1,\dots,F_n) = \frac{1}{Z}p(C)\prod_{i=1}^{n} p(F_i|C)$$

where Z is the normalization factor to ensure the distribution is a valid probability distribution.

3.5 tf.idf-SVM Classification algorithm:

The basic idea of the *tf.idf* algorithm is to represent each document *d* as a vector $\vec{d} = (d_1, d_2, \dots, d_{|F|})$ in a vector space so that documents with similar content have similar vectors. Each dimension of the vector space represents a word selected by the feature selection process.

The value of vector elements d_i for a document *d* is calculated as a product of the statistics *tf* (term frequency) and *idf* (inverse document frequency). Vector element d_i is calculated as follows:

$$d_i = tf(w_i, d)*idf(w_i)$$

where d_i is also called weight of word w_i in document *d*.

$$tf(w_i, d)= (n_i, d)/(N, d)$$

where (n_i, d) is the frequency of word w_i in document *d* and (N, d) is the number of words in document *d*.

$$idf(w_i)=log(|D|/df(w_i))$$

where $|D|$ is the number of documents and $df(w_i)$ is the number of documents containing word w_i.

A word w_i is an important indexing term for document *d* if it occurs frequently in it (the term frequency is high). On the other hand, words which occur in many documents are rated less important indexing terms due to their low inverse documents frequency.

SVM model:

Support Vector Machines (SVM) have been proven as one of the most powerful learning algorithms for text categorization [2].The SVM classifier toolkit we used in our project is Liblinear which is developed by Chih-Jen Lin of National Taiwan University (for the information of this classifier please see www.csie.ntu.edu.tw/~cjlin/liblinear).

We trained this classifier with training set to generate a model and used this model to predict the category of test document.

The toolkit contains two executable files(train.exe and predict.exe), which are used to generate the model and predict the target category of test document. We use cmd.exe in windows to call these two tools.

First, use our program to generate the vector space file, use train.exe to train the classifier and get the train.model.

Second, use predict.exe to predict test (vsm) in train model and generate the answer.

Because articles in Reuter's corpus often belong to more than one categories, if the prediction is one of the categories of test document, we treat the prediction is right.

3.6 Modified TFIDF Classification

We hold the point those words appear in more documents in certain category *C* should have higher weights. For example, in category "earn", words such as "profit", "income" and "tax" occur frequently. Suppose that the word "income" and "tax" have the same *tf* value, while "tax" appears in 20 documents and "income" appears in 35 documents in category "earn". We think that word "income" contributes more to classify the document to category "earn" and therefore should has higher weight. So we modified the idf equation as follows:

$$idf(\mathrm{w}_{ic}) = \log(\frac{|D|}{df(w_i)} * df(w_{ic}))$$

Where |D| is the number of documents in data set, $df(w_i)$ is the number of document containing word t_i and $df(w_{ic})$ is the number of documents in category *C* containing word t_i.

4 Result analysis

Table 1 Accuracy of different methods for document classification

Methods	Accuracy %
Frequency Counting Based (control)	53.8
Naive Bayes	78.8
tf.idf-SVM	84.0
Modified tf.idf-SVM	84.3

From the results we can see that Frequency Counting Based method has the lowest performance. It is a very primary method that the occurrence in a category of a word is value of its feature. This method assumes that all the words in a category contain the same amount of information and thus make little use of informative features.

All other methods make a great improvement compared to Frequency Counting Based method. Tf.idf-SVM method has better performance compared to NB model. It may be caused by *idf* assign those informative words ("realty", "enterprise") higher weights but lower the weights of those words with little information("average", "cover", "enter").

However, the result shows that the performance of modified tf.idf-SVM method doesn't improve as much as we expected. The accuracy of modified tf.idf-SVM method is 84.3%, which is only slightly higher than the original method, 84.0%.

Theoretically, apply the modification on the tf.idf algorithm would improve the accuracy of tf.idf-SVM method. But because this modification those words containing little amount of information, such as

"average", "cover" and "enter" higher weights than original method and thus reduce the efficiency of idf algorithm. We thought this is the reason why our modification slightly improves the accuracy of tf.idf-SVM method.

However, we still hold the point that this modification is meaningful because tf.idf algorithm doesn't consider giving those significant words which appears frequently in documents of the same class higher weight. We thought there is still improving space for our modified method.

5 Conclusion and Future Directions

In conclusion, we implemented and compared various classification methods on Reuter's corpus and found that all methods except Frequency Counting Based method performed well with accuracy above 78.8%. It is very exciting. Among these methods, the accuracy of tf.idf-SVM method is better than Naive Bayes method. We also attempted to modify the tf.idf-SVM method to improve its performance. Its accuracy showed a slight improvement and we attributed the reason why it didn't present obvious increase to the reduction of the efficiency of the idf.

As newcomers of Natural Language Process, this project is really a tough work for us. It is a shame that we just tried the low level methods at the beginning. Therefore we added something new to this project after the presentation.

For future work, while other techniques for document classification could be explored, we believe that there is plenty scope for future work with the methods discussed in this report.

For feature selection process, more complicated feature-sets could be examined. The feature-sets we explored provided a good base for classification, but all the features are individual words. More complicated features could be added to augment accuracy. For example bigrams or n-grams contain more information than individual word and thus have the potential to improve the accuracy. Feature-sets work best with articles in particular could also be explored more fully. In our project, we tried to add the weights of those words play a big part in distinguish between categories by modification of idf algorithm. This could also work in feature selection process.

For modified tf.idf-SVM method, like says previously, we focus more on the difference of words in one category. But how to efficiently discriminate those features is an interesting topic. In this paper, we use the number of documents to measure those features and we think there will be a better method.

1. Implement this model on other datasets to test the universality of this method.
2. Combine the tf.idf with other classifier method to find suitable model.
3. Change the new idf form to pursue better performance.

References

1. Farkas, J. *Generating document clusters using thesauri and neural networks*. in *Electrical and Computer Engineering, 1994. Conference Proceedings. 1994 Canadian Conference on*. 1994. IEEE.
2. Svingen, B. *Using Genetic Programming for Document Classification*. in *FLAIRS Conference*. 1998.

3. Cohen, W.W. and Y. Singer, *Context-sensitive learning methods for text categorization.* ACM Transactions on Information Systems (TOIS), 1999. **17**(2): p. 141-173.
4. Nigam, K., J. Lafferty, and A. McCallum. *Using maximum entropy for text classification*. in *IJCAI-99 workshop on machine learning for information filtering*. 1999.
5. Nigam, K., et al., *Text classification from labeled and unlabeled documents using EM*. Machine learning, 2000. **39**(2-3): p. 103-134.
6. Lewis, D.D., *Naive (Bayes) at forty: The independence assumption in information retrieval*, in *Machine learning: ECML-98*. 1998, Springer. p. 4-15.
7. McCallum, A. and K. Nigam. *A comparison of event models for naive bayes text classification*. in *AAAI-98 workshop on learning for text categorization*. 1998. Citeseer.
8. Joachims, T., *Text categorization with support vector machines: Learning with many relevant features*. 1998: Springer.
9. Manevitz, L.M. and M. Yousef, *One-class SVMs for document classification*. the Journal of machine Learning research, 2002. **2**: p. 139-154.
10. Wu, H.C., et al., *Interpreting tf-idf term weights as making relevance decisions.* ACM Transactions on Information Systems (TOIS), 2008. **26**(3): p. 13.
11. Salton, G. and C. Buckley, *Term-weighting approaches in automatic text retrieval.* Information processing & management, 1988. **24**(5): p. 513-523.

Appendix

1. Feature Selection Method of Information Gain:

The entropy of the whole dataset can be described as follows:

$$\mathrm{H(C)} = -\sum_{ci}^{C} p(c_i)\log(p(c_i))$$

Where *C* is the category set and $p(c_i)$ is probability of category set.

The entropy for feature t of category C can be described as follows:

$$H(C_i|t) = P(t)H(C_i|t) + P(\bar{t})H(C_i|\bar{t})$$

Where p(T) represents the probability of feature *t* in category i. $\mathrm{P}(\bar{t})$ represents for the probability of feature t not in category i

Combining H(s) and H(C|T), we can get the formula of Information Gain:

$$\mathrm{IG(T)} = \mathrm{H(C)} - \mathrm{H(C|T)}$$

$$= -\sum_{i=1}^{C} P(C_i)\ \log(p(C_i) + \ \mathrm{P(t)}\sum_{i=1}^{C} \mathrm{P}(C_i|\mathrm{t})\log(p(C_i|t) + \mathrm{P}(\bar{t})\sum_{i=1}^{C} \mathrm{P}(C_i|\bar{\mathrm{t}})\log(p(C_i|\bar{t})$$

And we rank all those words in IG value, select 20000 as the feature to classify the datasets.

2.Other Modification Method of TFIDF:

(1) Some change IDF as:

$$idf_i = \log\frac{|D|}{|\{d: t_i \in d\}|} * log\frac{\left|\{c_i: t_{ij} \in c_i\}\right|}{|C_i|}$$

$|D|$ is the number of all docs

$|\{d: t_i \in d\}|$ is the number of doc which contains the word t_i

$\left|\left|\{c_i: t_{ij} \in c_i\}\right|\right|$ the number of feature word which appears in class i.

$|C_i|$ represents the number of doc in certain class.

(2)Some change IDF as:

$$idf_i = \log(\frac{|D|}{|\{d: t_i \in d\}|} * \left|\{c_i: t_{ij} \in c_i\}\right|)$$

$|D|$ is the number of all docs

$|\{d: t_i \in d\}|$ is the number of doc which contains the word t_i

$\left|\left|\{c_i: t_{ij} \in c_i\}\right|\right|$ the number of feature word which appears in class i.

Those methods don't perform well on Reuters dataset (accuracy is only 77.3% and 74.1%). But we take use of their modification of the formula in our project.