Using language models for text classification

Using Language Models for Text Classification

Jing Bai, Jian-Yun Nie
Département d'informatique et de recherche opérationnelle
Université de Montréal
C.P. 6128, succursale Centre-ville
Montréal, Québec, H3C 3J7 Canada
{baijing, nie}@iro.umontreal.ca

ABSTRACT

This paper describes an approach to text classification using language models. This approach is a natural extension of the traditional Naïve Bayes classifier, in which we replace the Laplace smoothing by some more sophisticated smoothing methods. In this paper, we tested four smoothing methods commonly used in information retrieval. Our experimental results show that using a language model, we are able to obtain better performance than traditional Naïve Bayes classifier. In addition, we also introduce into the existing smoothing methods an additional factor of smoothing scale according to the amount of training data of the class, and this allows us to further improve the classification performance.

Keywords

language model, text classification, smoothing, Naïve Bayes

1. INTRODUCTION

Language models (LM) have been successfully applied in many application areas such as speech recognition and statistical natural language processing. Recently, a number of researches have confirmed that language model is also an effective and attractive approach for information retrieval (IR) [6, 11]. It not only provides an elegant theoretical framework to IR, but also results in effectiveness comparable to the best state of the art systems. This success has triggered a great interest of IR researchers, and LM has since been used to other IR-related tasks, such as topic detection and tracking [7]. However, until now, few attempts have been made to use language models for text classification. This paper describes our attempt to use LM in text classification.

Text classification aims to assign text documents into one or more predefined classes based on their contents. Many machine learning techniques have been applied to automatic text classification, such as Naïve Bayes (NB), K-Nearest Neighbor and Support Vector Machines (SVM). Although several experiments have shown that SVM can produce better classification results than NB, this latter is still widely used in text classification for its simplicity and efficiency [1]. It is still interesting to make further improvements on NB. Language modeling approaches provide interesting tools for this.

If we analyze the principle of Naïve Bayes and language models, we can clearly see a strong similarity between them: In fact, Naïve Bayes has to estimate the probability that a word appears in a class. This estimation is a form of construction of a language model. In particular, Naïve Bayes also uses a simple smoothing method – Laplace – to deal with the zero-probability problem. In other language models developed in speech recognition or IR, many more sophisticated smoothing methods have been developed. Therefore, it is natural to extend the current approach of Naïve Bayes to make use of other language models. In this paper, we investigate the utilization of language models to text classification. The experimental results show that our approach can improve the classification performance compared with the traditional Naïve Bayes classifier.

The paper will be organized as follows. In Section 2, we will briefly review the Naïve Bayes classifier and some related works. In Section 3, we describe our approach to text classification using language models. Section 4 and 5 present the experimental design and results on the Reuters-21578 data set. Some comparisons to the related research will also be shown. Finally, Section 7 gives some conclusions.

2. RELATED WORK

2.1 Naïve Bayes Classifier

Given a document *d* and a set of predefined classes {… c_i, …}, Naïve Bayes classifier first computes the posterior probability that the document belongs to each particular class c_i, i.e., $P(c_i \mid d)$, and then assigns the document to the class(es) with the highest probability value(s). The posterior probability is computed by applying the Bayes rule:

$$P(c_i \mid d) = \frac{P(d \mid c_i)P(c_i)}{P(d)} \qquad (1)$$

The denominator $P(d)$ in formula (1) is independent from classes; therefore, it can be ignored for the purpose of class ranking. Therefore:

$$P(c_i \mid d) \propto P(d \mid c_i)P(c_i) \qquad (2)$$

In Naïve Bayes, it is further assumed that words are independent given a class, i.e., for a document $d = d_1,\ldots,d_m$

$$P(d \mid c_i) = \prod_{j=1}^{m} P(d_j \mid c_i)$$

Formula (2) can then be simply expressed as follows:

$$P(c_i \mid d) \propto \prod_{j=1}^{m} P(d_j \mid c_i) P(c_i) \qquad (3)$$

In formula (3), $P(c_i)$ can be estimated by the percentage of the training examples belonging to class c_i:

$$P(c_i) = \frac{N_i}{N}$$

where N is the total number of training documents and N_i is the number of training documents in class c_i. $P(d_j \mid c_i)$ is usually determined by:

$$P(d_j \mid c_i) = \frac{1 + count(d_j, c_i)}{|V| + N_i}$$

where $count(d_j, c_i)$ is the number of times that word d_j occurs within the training documents of class c_i, and $|V|$ is the total number of vocabulary. This estimation uses the Laplace (or add-one) smoothing to solve the zero-probability problem.

Despite the simplicity of Naïve Bayes classifier, namely due to the independence assumption, this classifier is surprisingly effective [1]. It is still a representative method in the current state of the art of text classification.

2.2 Language Modeling Approach in IR

Language modeling has been applied successfully in information retrieval [6, 11, 12] and several related applications such as topic detection and tracking [7]. Given a document *d* and a query *q*, the basic principle of this approach is to compute the conditional probability $P(d \mid q)$ as follows:

$$P(d \mid q) = \frac{P(q \mid d) P(d)}{P(q)} \propto P(q \mid d) P(d)$$

If we assume $P(d)$ to be a constant, then the ranking of a document *d* for a query *q* is determined by $P(q \mid d)$. The calculation of this value is performed as follows: we first construct a statistical language model $P(. \mid d)$ for the document *d*, called document model. Then $P(q \mid d)$ is estimated as the probability that the query can be generated from the document model. This probability is often calculated by making the assumption that words are independent (in a unigram model) in a similar way to Naïve Bayes. This means that for a query $q = q_1, \ldots, q_n$, we have:

$$P(q \mid d) = \prod_{j=1}^{n} P(w_j \mid d)$$

In the previous studies, it turns out that smoothing is a very important process in building a language model [11]. The effectiveness of a language model approach is strongly dependent on the way that the document language model is smoothed. The primary goal of smoothing is to assign a non-zero probability to the unseen words and to improve the maximum likelihood estimation. However, in IR application, smoothing also allows us to consider the global distribution of terms in the whole collection, i.e., the *idf* factor used in IR [11].

Several different smoothing methods such as Dirichlet, Absolute discount, etc. have been applied in language models. In Zhai and Lafferty [11], it has been found that the retrieval effectiveness is generally sensitive to the smoothing parameters. In our experiments on classification, we also observed similar effects.

3. USING LANGUAGE MODELS FOR TEXT CLASSIFICATION

If we compare Naïve Bayes with the general language modeling approach in IR, we can observe a remarkable similarity: the general probabilistic framework is the same, and both use smoothing to solve the zero-probability problem. The difference between them lies in the objects which a language model is constructed for and applied to. In IR, one builds a LM for a document and applies it to a query, whereas in NB classifier, one builds a LM for a class and applies it to a document. However, we also observe that in the implementation of NB, one usually is limited to the Laplace smoothing. Few attempts have been made in using more sophisticated smoothing techniques, except a recent study by Peng et al. [5]. In this study, Peng et al. used several smoothing techniques developed in statistical language modeling for text classification.

As the experiments in IR showed, the effectiveness of language model strongly depends on the smoothing techniques, and several smoothing techniques have proven to be effective. Then a natural question is whether it is also beneficial in classification to use other sophisticated smoothing methods instead of the Laplace smoothing. In this paper, we will focus on this problem. As we will see later in our experiments, it will be clear that such a replacement can bring improvements to Naïve Bayes classifier.

3.1 Principle

The basic principle of our approach to text classification using language models is straightforward.

As in Naïve Bayes, the score of a class c_i for a given document *d* is estimated by formula (3). However, the estimation of $P(d_j \mid c_i)$ is different: it will be estimated from the language modeling perspective. First, we construct a language model for each class with several smoothing techniques. Then $P(d_j \mid c_i)$ is the probability that the term d_j can be generated from this model. As smoothing turns out to be crucial in IR experiments, it is also necessary to carefully select the smoothing methods. In the next section, we will describe those that have been used in several experiments in IR.

3.2 Smoothing Techniques for Estimation

A number of smoothing techniques have been developed in statistical natural language processing to estimate the probability of a word or an n-gram. As we mentioned earlier, the primary goal is to attribute a non-zero probability to the words or n-grams that are not seen in a set of training documents. Two basic ideas have been used in smoothing: 1) using a lower-order model to supplement a higher-order model; 2) modifying the frequency of word occurrences.

In IR, both ideas have been used. On the first solution, it is common in IR to utilize the whole collection of documents to construct a background model. This model is considered as a lower-order model to the document model, although both models may be unigram models. This solution has been useful for relatively short documents. Although a class usually contains more than one document, thus longer than a single document, the same problem of imprecise estimation exists, especially for small classes. Therefore, one can use the same approach of smoothing to classification. The second solution is often used in combination with the first one (i.e., one simultaneously use the collection model and change the word counts), as we can see in the smoothing methods described below.

Two general formulations are used in smoothing: backoff and interpolation. Both smoothing methods can be expressed in the following general form [12]:

$$P(w|c_i) = \begin{cases} P_s(w|c_i) & \text{w is seen in } c_i \\ \alpha_{c_i} P_u(w|C) & \text{w is unseen in } c_i \end{cases}$$

That is, for a class c_i, one estimate is made for the words seen in the class, and another estimate is made for the unseen words. In the second case, the estimate for unseen words is based on the entire collection, i.e., the collection model. The effect of incorporating the collection model not only allows us solving the zero-probability problem, but also is a way to produce the same effect as the *idf* factor commonly used in IR (as shown in [11]).

In our experiments, we tested the following specific smoothing methods:

- Jelinek-Mercer (JM) smoothing:

$$P_{JM}(w|c_i) = (1-\lambda)P_{ml}(w|c_i) + \lambda P(w|C)$$

which linearly combines the maximum likelihood estimate $P_{ml}(w|c_i)$ of the class model with an estimate of the collection model.

- Dirichlet smoothing:

$$P_{Dir}(w|c_i) = \frac{c(w,c_i) + \mu P(w|C)}{|c_i| + \mu}$$

where $c(w,c_i)$ is the count of word w in c_i, $|c_i|$ is the size of c_i (i.e., the total word count of c_i) and μ is a pseudo-count.

- Absolute discount smoothing:

$$P_{AD}(w|c_i) = \frac{\max(c(w,c_i) - \delta, 0) + \delta |c_i|_u \ P(w|C)}{|c_i|}$$

in which the count of each word is reduced by a constant $\delta \in [0, 1]$ and the discounted probability mass is redistributed on the unseen words proportionally to their probability in the collection model. In the above equation, $|c_i|_u$ is the number of unique words in c_i.

- Two-stage (TS) smoothing [12]:

$$P_{TS}(w|c_i) = (1-\lambda)\frac{c(w,c_i) + \mu P(w|C)}{|c_i| + \mu} + \lambda P(w|C)$$

This smoothing method combines Dirichlet smoothing with an interpolation smoothing.

In the previous experiments of IR, it turns out that Dirichlet and two-stage smoothing methods provided very good effectiveness. In our experiments, we will test whether these smoothing methods, when applied to text classification, bring similar impact.

4. CORPUS AND PERFORMANCE MEASURE

4.1 Corpus

In order to compare with the previous results, our experiments have been conducted on the benchmark corpus of Reuters-21578, gathered from Reuter's newswire articles. We chose the ModApte split of Reuters-21578 data set, which is commonly used for text classification research today [9]. There are 135 topic classes, but we used only those 90 for which there exists at least one document in both the training and test set. Then we obtained 7769 training documents and 3019 test documents. The number of training documents per class varies from 2877 to 1. The largest 10 classes contain 75% of the documents, and 33% classes have fewer than 10 training documents.

4.2 Performance Measure

For the purpose of comparison with the previous researches, we evaluate the performance of classification in terms of standard recall, precision and F_1 measure. For evaluating average performance across classes, we used micro-averaging and macro-averaging. Micro-averaging scores are the scores calculated by mixing together the documents across all the classes. Macro-averaging scores are the averages of the scores of each class calculated separately. Micro-averaging gives an equal weight to every document, thus putting more emphasis on larger classes. On the other hand, macro-averaging gives an equal weight to every class regardless how rare or how common a class is. In [9], it is claimed that micro-averaging can better reflect the real classification performance than macro-averaging. Therefore, our observations will be made mainly on micro-averaging.

5. EXPERIMENTAL EVALUATION

5.1 Naïve Bayes Classifier

To provide the comparable results of classification on Reuters-21578 corpus, we used the multinomial mixture model of Naïve Bayes classifier of the Rainbow package, developed by McCallum [3].

In Naïve Bayes classifier, feature selection is important. The effect of feature selection is to remove meaningless features (words) so that classification can be determined according to meaningful features. Several feature selection methods are commonly used: information gain (IG), chi-square, mutual information, etc. Information gain has shown to produce good results in [9]. The information gain of a word w is calculated as follows:

$$IG(w) = -\sum_{i=1}^{k} P(c_i)\log P(c_i) + P(w)\sum_{i=1}^{k} P(c_i \mid w)\log P(c_i \mid w) + P(\overline{w})\sum_{i=1}^{k} P(c_i \mid \overline{w})\log P(c_i \mid \overline{w})$$

where $\overline{w}$ means the absence of the word *w*.

One can choose a fixed number of features according to their IG, or set up a threshold on IG to make the selection. The following table shows the classification results by NB without feature selection and with a selection of 2000 features according to IG. The number 2000 is suggested in [9].

NB	miR	miP	miF$_1$	maF$_1$	Error
all features Rcut$_1$	0.6990	0.8668	0.7739	0.1838	0.00563
2K features Rcut$_1$	0.7145	0.8861	0.7911	0.3594	0.00520
2K features Scut	0.7254	0.8885	0.7987	0.3675	0.00504

miR: micro-avg recall miP: micro-avg precision

miF$_1$: micro-avg F$_1$ maF$_1$: macro-avg F$_1$

Table 1 Performance of NB on Reuters-21578

The final step of classification is to assign (or select) one or more classes to each document. The two first lines of Table 1 have been obtained with the simplest class selection method: Rcut (or Rank-based cut). We select the first class suggested by the classifier. One can also select more than one class by this method, according to the average number of classes a document belongs to. For the Reuters collection, selecting only the first class gives the best results. Besides this selection method, several other methods have also been proposed: Pcut and Scut [10]. Pcut (proportional cut) makes the selection for each class (instead of for each document in Rcut) according to the size of the class, i.e., the number of training documents in the class: The larger a class is, the more documents will be classified in it. Scut (score-based local optimization) tries to determine the best threshold value of the normalized score to be used for the selection for each class. Typically, one uses a held-out from the training documents to tune the threshold. In our experiments, we will only use Rcut and Scut.

5.2 Language Modeling Approach

In the experiments using language models, we used the Lemur toolkit, which is designed and developed by Carnegie Mellon University and the University of Massachusetts [2]. The system allows us to train a language model for each class using a set of training documents, and to calculate the likelihood of a document according to each class model, i.e., $P(d \mid c_i)$. The final score of a class can then be computed according to formula 2.

Different Smoothing Methods

In our experiments, we used the four smoothing methods that are described earlier by varying the parameters. The following table shows the results by each method obtained with Rcut$_1$. No feature selection is made. The percentages in the table are the relative changes with respect to NB with no feature selection (Table 1).

Smoothing	miR	miP	miF$_1$	maF$_1$	Error
Jelinek-Mercer (λ=0.31)	0.7078	0.8778	0.7837 (+1.3%)	0.4659 (+153.5%)	0.00538
Dichichlet (μ=9500)	0.7051	0.8745	0.7807 (+0.9%)	0.3986 (+116.9%)	0.00546
Absolute (δ=0.83)	0.7118	0.8827	0.7881 (+1.8%)	0.4839 (+163.3%)	0.00527
Two-stage (λ=0.86,μ=6000)	0.7260	0.9003	0.8038 (+3.9%)	0.4214 (+129.3%)	0.00488

Table 2 Performance of LM on Reuters-21578 with Rcut$_1$, no feature selection

Globally, our experiments show that using language model may improve classification effectiveness over Naïve Bayes. This is true especially for macro-averaging F$_1$ which is much higher than with NB. The improvement on micro-averaging F$_1$ is less obvious. The three first smoothing methods only lead to marginal improvements over NB. On the other hand, two-stage smoothing produces a larger improvement on micro-averaging F$_1$ over NB. The comparison of the improvements on micro- and macro-averaging F1 seems to suggest that language models can bring larger improvements to small classes than to large classes. A possible reason is that our smoothing methods also combine the collection probabilities, instead of only changing the frequencies of words as in NB (Laplace smoothing). By modifying the frequency of words in Laplace smoothing, all the unseen words, either meaningful or not, will be attributed an equal probability. However, the smoothing methods with the collection model attribute different probabilities to unseen words according to their global distribution in the collection. Therefore, the latter probabilities can better reflect the characteristics of the collection and of the language. In our experiments, the addition of the collection model seems to benefit greatly small classes with less training data for which a heavy smoothing is required.

We also observe that the classification performance highly dependent on the choice of a smoothing method and its parameters. This observation is similar to that of Zhai and Lafferty [11] for IR.

In order to test statistical significance of the changes of performance, Yang and Liu [9] proposed several statistical tests. Here, we use the micro t-test, which compares paired F1 values obtained for each class. In turns out that all the improvements obtained with the four smoothing methods are statistically significant, with p-values < 0.001[1].

Feature Selection with Language Model

Feature selection has been very useful for NB classifier. Would it produce a similar effect on language models? In order to answer this question, we conducted a series of experiments using different numbers of features selected according to information gain. The following table shows the results of doing feature selection on the four smoothing methods shown in Table 2.

[1] A p-value lower than 0.05 is considered to be statistically significant at the 0.05 significance level.

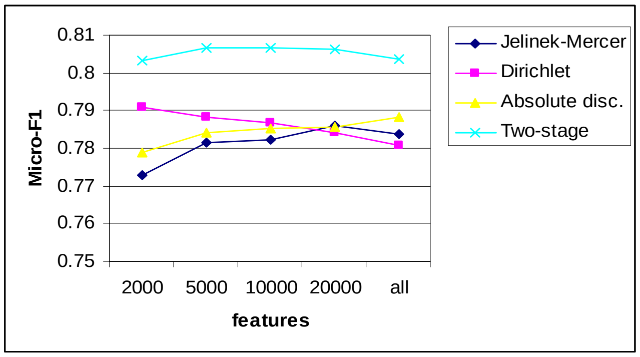

Figure 1 The effects of feature selection on different smoothing methods

These results do not show any performance improvement when we use feature selection, except for Dirichlet smoothing. This conclusion seems contradictory to the results with NB, and counter-intuitive at the first glance. However, one can possibly explain this by the fact that, as the class model has been massively smoothed with the collection model; those non-discriminative features do not make a significant difference between documents with respect to a class. Therefore, the inclusion of such features in the calculation of the score does not hurt as much as in NB, which does not incorporate the collection model. This suggests that the consideration of the collection model in smoothing renders feature selection less necessary. A similar conjecture has been made in [5].

Class Selection

This experiment aims to verify whether Scut also improves classification performance with language models. For Scut, we used 10% held-out documents from each class selected randomly to determine the optimal threshold on normalized score (divided by the maximal score), and this process is carried out 10 times. The final optimal threshold for each class is the average of the 10 trials.

Table 3 shows the comparisons between Scut and Rcut with the two-stage smoothing (λ=0.86, μ=6000).

Twostage	miR	miP	miF$_1$	maF$_1$	Error
Rcut$_1$	0.7260	0.9003	0.8038	0.4214	0.00488
Scut	0.7567	0.8688	0.8089	0.4469	0.00493

Table 3 Comparison of Rcut and Scut with Two-stage smoothing (λ=0.86, μ=6000)

This comparison shows no change in micro-averaging F1 using Scut in this case.

5.3 Varying Smoothing Scale

In an interpolation smoothing method, such as Jelinek-Mercer or Two-stage smoothing, the coefficient λ determines the scale in which the collection model is used to complement the document model. We notice that this coefficient usually takes a fixed value in the previous experiments.

Intuitively, in IR, for a shorter document, there is a stronger need for smoothing with the collection model than a long document. Therefore, the coefficient λ should be higher for a short document than for a long document. For classification, the same intuition applies: the class model for a large class which contains a large number of training documents has a lower requirement for smoothing, than a small class with few training documents. Based on this intuition, we add another scaling coefficient $I_i = 1 - \frac{|c_i|}{|C|}$ (where $|c_i|$ and $|C|$ are respectively the size of the class and of the training collection in number of words) into the formulas to vary the smoothing scale according to the size of the class as follows:

- Scaled Jelinek-Mercer (SJM) smoothing:
$$P_{SJM}(w|c_i) = (1-\lambda\times I_i)P_{ml}(w|c_i) + \lambda\times I_i \times P(w|C)$$
- Scaled Two-Stage (STS) smoothing:
$$P_{STS}(w|c_i) = (1-\lambda\times I_i)\frac{c(w,c_i)+\mu P(w|C)}{|c_i|+\mu} + \lambda\times I_i \times P(w|C)$$

The following table shows a comparison of the best classification performance with the original and scaled smoothing methods.

Smoothing	miR	miP	miF$_1$	maF$_1$	Error
JM (λ=0.31)	0.7078	0.8778	0.7837	0.4659	0.00538
SJM (λ=0.95)	0.7212	0.8943	0.7985 (+1.9%)	0.5085 (+9.1%)	0.00502
TS (λ=0.86,μ=6000)	0.7260	0.9003	0.8038	0.4214	0.00488
STS (λ=0.9,μ=3500)	0.7318	0.9076	0.8103 (+0.8%)	0.4347 (+3.2%)	0.00472

Table 4 Comparison between smoothing methods and their modified versions

We observe that for these two smoothing methods, the addition of the I_i coefficient is helpful: for both cases, some improvements are obtained.

As we can also observe, the best performance values are obtained with different values of λ with or without the additional I_i coefficient. This is expectable, because the actual smoothing scale is determined by $\lambda \times I$, and we have different values for different classes. For example, $\lambda \times I$ for the largest class ("earn") would be 0.665 in SJM, whereas it is 0.95 for the smallest classes.

However, the macro t-tests have not shown that the changes using the scaled versions of smoothing are statistically significant (with p = 0.0628 and p = 0.3291 respectively for SJM and STS).

If we further use Scut instead of Rcut$_1$, the performances can be further improved. This is shown in the following table.

Smoothing	miR	miP	miF$_1$	maF$_1$	Error
SJM Scut	0.7684	0.8560	0.8099	0.5265	0.00497
STS Scut	0.7631	0.8788	0.8169	0.4470	0.00471

Table 5 Improvement of SJM and STS with Scut

If we compare these results with the best performance obtained for NB (miF$_1$ = 0.7987), we can conclude that the improvement using STS are relatively large (2.3%), and this is statistically significant (p = 0.0016).

6. COMPARISON TO RELATED RESEARCH

The language modeling approach has been applied to text classification recently [5]. In the experiments of Peng et al., several other smoothing techniques, such as Good-Turing, Witten-Bell, have been used. The general framework is similar to ours. However, Peng et al. used perplexity to rank the candidate classes, whereas we use document likelihood. In addition, Peng et al. assumed a uniform class prior $P(c)$, and found that there is little experimental difference between this prior and an empirically estimated prior. In our case, we use the latter, i.e., $P(c)$ is estimated by the proportion of the number of documents in c compared with the entire training collection. Our experiments showed that this prior can produce better results than the uniform prior.

Peng et al. also used bigram models for text classification, and this is shown to produce better results than unigram models in their tests. In contrast, the use of bigram models in IR has often produced marginal improvements over unigram models. This shows that the same language models may have different impact in IR and classification.

Despite the improvements we obtained with the two-stage smoothing over NB, the overall performance is still lower than one of the best classifiers – SVM. Using SVM, Yang reported a micro-averaging F_1 of 0.8599 on the same test data [9]. However, the current study is only a preliminary exploration. Further improvements can be brought to the utilization of language models later.

7. CONCLUSION

Language models seem to be a natural extension of Naïve Bayes because both use the same theoretical framework. The main difference between them lies in a stronger smoothing in the latter.

In this paper, we described several experiments using the language models developed in IR for text classification. Our experiments on Reuters-21578 data collection have shown noticeable improvements over NB, especially on the macro-averaging F1. On micro-averaging F1, we also observed some improvements, although they are lesser than on macro-averaging F1.

The best smoothing method among those tested is the two-stage smoothing. Our conclusion for classification using this method is similar to that in IR: two-stage smoothing is an effective smoothing method for text classification.

Despite the relatively modest improvements on micro-averaging F1, this preliminary study did show that language models can be used for text classification as a reasonable replacement of NB.

This study is limited to the utilization of unigram models. We are currently investigating the integration of bigram language models for text classification.

ACKNOWLEDGMENT

This work has been carried out within a joint research project with Nstein technologies. We would like to thank Nstein and NSERC for their support. We also thank François Paradis for providing the thresholding programs.

REFERENCES

[1] S. T. Dumais, J. Platt, D. Heckerman and M. Sahami (1998). Inductive learning algorithms and representations for text categorization. *In Proceedings of ACM-CIKM98*, Nov. 1998, pp. 148-155.

[2] The lemur toolkit for language modeling and information retrieval. http://www-2.cs.cmu.edu/~lemur

[3] A. McCallum (1996). Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/~mccallum/bow

[4] A. McCallum and K. Nigam (1998). A comparison of event models for Naïve Bayes text classification. *In Proceedings of AAAI-98 Workshop*, AAAI Press.

[5] F. Peng and D. Schuurmans (2003). Combining Naive Bayes and n-gram language models for text classification. *In Proceedings of the 25th European Conference on Information Retrieval Research (ECIR03)*, pp. 335-350.

[6] J. Ponte and W. B. Croft (1998). A language modeling approach to information retrieval. *In Proceedings of SIGIR 1998*. pp. 275-281.

[7] M. Spitters and W. Kraaij (2001), TNO at TDT2001: language model-based topic detection, *In Proceedings of Topic Detection and Tracking (TDT) Workshop 2001*.

[8] Y. Yang (1999). An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, Vol. 1, No. 1/2, pp. 67–88.

[9] Y. Yang and X. Liu (1999). A re-examination of text categorization methods. *In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 42-49.

[10] Y. Yang (2001). A study on thresholding strategies for text categorization. *In Proceedings of SIGIR 2001*, pp 137-145.

[11] C. Zhai and J. Lafferty (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. *In Proceedings of SIGIR 2001*, pp. 334-342.

[12] C. Zhai and J. Lafferty (2002). Two-stage language models for information retrieval. *In Proceeding of SIGIR 2002.*