# Text Classification in Asian Languages without Word Segmentation

**Fuchun Peng**[†‡] **Xiangji Huang**[†] **Dale Schuurmans**[†] **Shaojun Wang**[†⋆]
[†]School of Computer Science, University of Waterloo, Ontario, Canada
[‡]Department of Computer Science, University of Massachusetts, Amherst, MA, USA
[⋆]Department of Statistics, University of Toronto, Ontario, Canada
{f3peng, jhuang, dale, sjwang}@ai.uwaterloo.ca

## Abstract

We present a simple approach for Asian language text classification without word segmentation, based on statistical $n$-gram language modeling. In particular, we examine Chinese and Japanese text classification. With character $n$-gram models, our approach avoids word segmentation. However, unlike traditional ad hoc $n$-gram models, the statistical language modeling based approach has strong information theoretic basis and avoids explicit feature selection procedure which potentially loses significantly amount of useful information. We systematically study the key factors in language modeling and their influence on classification. Experiments on Chinese TREC and Japanese NTCIR topic detection show that the simple approach can achieve better performance compared to traditional approaches while avoiding word segmentation, which demonstrates its superiority in Asian language text classification.

## 1 Introduction

Text classification addresses the problem of assigning a given passage of text (or a document) to one or more predefined classes. This is an important area of information retrieval research that has been heavily investigated, although most of the research activity has concentrated on English text (Dumais, 1998; Yang, 1999). Text classification in Asian languages such as Chinese and Japanese, however, is also an important (and relatively more recent) area of research that introduces a number of additional difficulties. One difficulty with Chinese and Japanese text classification is that, unlike English, Chinese and Japanese texts do not have explicit whitespace between words. This means that some form of word segmentation is normally required before further processing. However, word segmentation itself is a difficult problem in these languages. A second difficulty is that there is a lack of standard benchmark data sets for these languages. Nevertheless, recently, there has been significant notable progress on Chinese and Japanese text classification (Aizawa, 2001; He et al., 2001).

Many standard machine learning techniques have been applied to text categorization problems, such as naive Bayes classifiers, support vector machines, linear least squares models, neural networks, and k-nearest neighbor classifiers (Sebastiani, 2002; Yang, 1999). Unfortunately, most current text classifiers work with word level features. However, word identification in Asian languages, such as Chinese and Japanese, is itself a hard problem. To avoid the word segmentation problems, character level $n$-gram models have been proposed (Cavnar and Trenkle, 1994; Damashek, 1995). There, they used $n$-grams as features for a traditional feature selection process and then deployed classifiers based on calculating feature-vector similarities. This approach has many shortcomings. First, there are an enormous number of possible features to consider in text categorization, and standard feature selection approaches do not always cope well in such circumstances. For example, given a sufficiently large number of features, the cumulative effect of uncommon features can still have an important effect on classification accuracy, even though infrequent features contribute less information than common features individually. Therefore, throwing away uncommon features is usually not an appropriate strategy in this domain (Aizawa, 2001). Another problem is that feature selection normally uses indirect tests, such as $\chi^2$ or mutual information, which involve setting arbitrary thresholds and conducting a heuristic greedy search to find a good subset of features. Moreover, by treating text categorization as a classical classification problem, standard approaches can ignore the fact that texts are written in natural language, which means that they have many implicit regularities that can be well modeled by specific tools from natural language processing.

In this paper, we present a simple text categorization approach based on statistical $n$-gram language modeling to overcome the above shortcomings in a principled fashion. An advantage we exploit is that the language modeling approach does not discard low frequency features during classification, as is commonly done in traditional classification learning approaches. Also, the language modeling approach uses $n$-gram models to capture more contextual information than standard bag-of-words approaches, and employs better smoothing techniques than standard classification learning. These advantages are supported by our empirical results on Chinese and Japanese data.

## 2 Language Model Text Classifiers

The goal of language modeling is to predict the probability of natural word sequences; or more simply, to put high probability on word sequences that actually occur (and low probability on word sequences that never occur). Given a word sequence $w_1 w_2 ... w_T$ to be used as a test corpus, the quality of a language model can be measured by the empirical perplexity (or entropy) on this corpus

$$\textit{Perplexity} = \sqrt[T]{\prod_{i=1}^{T} \frac{1}{P(w_i|w_1...w_{i-1})}} \quad (1)$$

The goal of language modeling is to obtain a small perplexity.

### 2.1 $N$-gram language modeling

The simplest and most successful basis for language modeling is the $n$-gram model. Note that by the chain rule of probability we can write the probability of any sequence as

$$P(w_1 w_2 ... w_T) = \prod_{i=1}^{T} P(w_i|w_1...w_{i-1}) \quad (2)$$

An $n$-gram model approximates this probability by assuming that the only words relevant to predicting $P(w_i|w_1...w_{i-1})$ are the previous $n-1$ words; that is, it assumes the Markov $n$-gram independence assumption

$$P(w_i|w_1...w_{i-1}) = P(w_i|w_{i-n+1}...w_{i-1})$$

A straightforward maximum likelihood estimate of $n$-gram probabilities from a corpus is given by the observed frequency

$$P(w_i|w_{i-n+1}...w_{i-1}) = \frac{\#(w_{i-n+1}...w_i)}{\#(w_{i-n+1}...w_{i-1})} \quad (3)$$

where #(.) is the number of occurrences of a specified gram in the training corpus. Unfortunately, using grams of length up to $n$ entails estimating the probability of $W^n$ events, where $W$ is the size of the word vocabulary. This quickly overwhelms modern computational and data resources for even modest choices of $n$ (beyond 3 to 6). Also, because of the heavy tailed nature of language (i.e. Zipf's law) one is likely to encounter novel $n$-grams that were never witnessed during training. Therefore, some mechanism for assigning non-zero probability to novel $n$-grams is a central and unavoidable issue. One standard approach to smoothing probability estimates to cope with sparse data problems (and to cope with potentially missing $n$-grams) is to use some sort of back-off estimator

$$P(w_i|w_{i-n+1}...w_{i-1}) = \begin{cases} \hat{P}(w_i|w_{i-n+1}...w_{i-1}), \\ \quad \text{if } \#(w_{i-n+1}...w_i) > 0 \\ \beta(w_{i-n+1}...w_{i-1}) \times P(w_i|w_{i-n+2}...w_{i-1}), \\ \quad \text{otherwise} \end{cases} \quad (4)$$

where

$$\hat{P}(w_i|w_{i-n+1}...w_{i-1}) = \frac{discount\,\#(w_{i-n+1}...w_i)}{\#(w_{i-n+1}...w_{i-1})} \tag{5}$$

is the discounted probability, and $\beta(w_{i-n+1}...w_{i-1})$ is a normalization constant calculated to be

$$\beta(w_{i-n+1}...w_{i-1}) = \frac{1 - \sum_{x:\#(w_{i-n+1}...w_{i-1}x)>0} \hat{P}(x|w_{i-n+1}...w_{i-1})}{1 - \sum_{x:\#(w_{i-n+1}...w_{i-1}x)>0} \hat{P}(x|w_{i-n+2}...w_{i-1})} \tag{6}$$

The discounted probability (5) can be computed using different smoothing approaches including Laplace smoothing, linear smoothing, absolute smoothing, Good-Turing smoothing and Witten-Bell smoothing (Chen and Goodman, 1998).

The language models described above use individual words as the basic unit, although one could instead consider models that use individual *characters* as the basic unit. The remaining details remain the same in this case. The only difference is that the character vocabulary is always much smaller than the word vocabulary, which means that one can normally use a much higher order, $n$, in a character level $n$-gram model (although the text spanned by a character model is still usually less than that spanned by a word model). The benefits of the character level model in the context of text classification are multi-fold: it avoids the need for explicit word segmentation in the case of Asian languages, and it greatly reduces the sparse data problems associated with large vocabulary models. In this paper, we experiment with character level models to avoid word segmentation in Chinese and Japanese.

### 2.2 Language models as text classifiers

Text classifiers attempt to identify attributes which distinguish documents in different categories. Such attributes may include vocabulary terms, word average length, local $n$-grams, or global syntactic and semantic properties. Language models also attempt capture such regularities, and hence provide another natural avenue to constructing text classifiers.

Our approach to applying language models to text categorization is to use Bayesian decision theory. Assume we wish to classify a text $d = w_1 w_2 ....w_N$ into a category $c \in C = \{c_1, ..., c_{|C|}\}$. A natural choice is to pick the category $c$ that has the largest posterior probability given the text. That is,

$$c^* = \arg\max_{c \in C} \Pr(c|d) \tag{7}$$

Using Bayes rule, this can be rewritten as

$$\begin{aligned} c^* &= \arg\max_{c \in C} \Pr(c) \Pr(d|c) && (8) \\ &= \arg\max_{c \in C} \Pr(c) \prod_{i=1}^{N} Pr_c(w_i|w_{i-n+1}...w_{i-1}) && (9) \end{aligned}$$

Here, $\Pr(d|c)$ is the likelihood of $d$ under category $c$, which can be computed by $n$-gram language modeling. The likelihood is related to perplexity by Equ. (1). The prior $\Pr(c)$ can be computed from training data or can be used to incorporate more assumptions, such as a uniform or Dirichelet distribution.

Therefore, our approach is to learn a separate back-off language model for each category, by training on a data set from that category. Then, to categorize a new text $d$, we supply $d$ to each language model, evaluate the likelihood (or entropy) of $d$ under the model, and pick the winning category according to Equ. (9).

The inference of an $n$-gram based text classifier is very similar to a naive Bayes classifier (to be dicussed below). In fact, $n$-gram classifiers are a straightforward generalization of naive Bayes (Peng and Schuurmans, 2003).

## 3 Traditional Text Classifiers

We introduce the three standard text classifiers that we will compare against below.

### 3.1 Naive Bayes classifiers

A simple yet effective learning algorithm for text classification is the naive Bayes classifier. In this model, a document $d$ is normally represented by a vector of $K$ attributes $d = (v_1, v_2, ....v_K)$. The naive Bayes model assumes that all of the attribute values $v_j$, are independent given the category label

$c$. Thus, a maximum a posteriori (MAP) classifier can be constructed as follows.

$$c^* = \arg\max_{c \in C} \left\{ P(c) \times \prod_{j=1}^{K} P(v_j|c) \right\} \qquad (10)$$

To cope with features that remain unobserved during training, the estimate of $P(v_j|c)$ is usually adjusted by Laplace smoothing

$$P(v_j|c) = \frac{N_j^c + a_j}{N^c + a} \qquad (11)$$

where $N_j^c$ is the frequency of attribute $j$ in $D^c$, $N^c = \sum_j N_j^c$, and $a = \sum_j a_j$. A special case of Laplace smoothing is *add one* smoothing, obtained by setting $a_j = 1$. We use add one smoothing in our experiments below.

### 3.2 Ad hoc $n$-gram text classifiers

In this method a test document $d$ and a class label $c$ are both represented by vectors of $n$-gram features, and a distance measure between the representations of $d$ and $c$ is defined. The features to be used during classification are usually selected by employing heuristic methods, such as $\chi^2$ or mutual information scoring, that involve setting cutoff thresholds and conducting a greedy search for a good feature subset. We refer this method as ad hoc $n$-gram based text classifier. The final classification decision is made according to

$$c^* = \arg\min_{c \in C} \{\text{distance}(d, c)\} \qquad (12)$$

Different distance metrics can be used in this approach. We implemented a simple re-ranking distance, which is sometimes referred to as the out-out-place (OOP) measure (Cavnar and Trenkle, 1994). In this method, a document is represented by an $n$-gram profile that contains selected $n$-grams sorted by decreasing frequency. For each $n$-gram in a test document profile, we find its counterpart in the class profile and compute the number of places its location differs. The distance between a test document and a class is computed by summing the individual out-of-place values.

### 3.3 Support vector machine classifiers

Given a set of $N$ linearly separable training examples $S = \{x_i \in R^n | i = 1, 2, ..., N\}$, where each sample belongs to one of the two classes, $y_i \in \{+1, -1\}$, the SVM approach seeks the optimal hyperplane $w \cdot x + b = 0$ that separates the positive and negative examples with the largest margin. The problem can be formulated as solving the following quadratic programming problem (Vapnik, 1995).

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}||w||^2 \\ \text{subject to} & y_i(w \cdot x_i + b) \geq 1 \end{array} \qquad (13)$$

In our experiments below, we use the $SVM^{light}$ (Joachims, 1998) toolkit with default settings.

## 4 Empirical evaluation

We now present our experimental results on Chinese and Japanese text classification problems. The Chinese data set we used has been previously investigated in (He et al., 2001). The corpus is a subset of the TREC-5 *People's Daily news corpus* published by the Linguistic Data Consortium (LDC) in 1995. The entire TREC-5 data set contains 164,789 documents on a variety of topics, including international and domestic news, sports, and culture. The corpus was originally intended for research on information retrieval. To make the data set suitable for text categorization, documents were first clustered into 101 groups that shared the same headline (as indicated by an SGML tag). The six most frequent groups were selected to make a Chinese text categorization data set.

For Japanese text classification, we consider the Japanese text classification data investigated by (Aizawa, 2001). This data set was converted from the NTCIR-J1 data set originally created for Japanese text retrieval research. The conversion process is similar to Chinese data. The final text classification dataset has 24 categories which are unevenly distributed.

### 4.1 Experimental paradigm

Both of the Chinese and Japanese data sets involve classifying into a large number of categories, where each document is assigned a single category. Many classification techniques, such as SVMs, are intrinsically defined for two class problems, and have to be extended to handle these multiple category data sets. For SVMs, we employ a standard technique of first converting the $|C|$ category classification problem to $|C|$ binary classification problems.

For the experiments on Chinese data, we follow (He et al., 2001) and convert the problem into 6 binary classification problems. In each case, we randomly select 500 positive examples and then select 500 negative examples evenly from among the remaining negative categories to form the training data. The testing set contains 100 positive documents and 100 negative documents generated in the same way. The training set and testing set do no overlap and do not contain repeated documents.

For the experiments on Japanese data, we follow (Aizawa, 2001) and directly experiment with a 24-class classification problem. The NTCIR data sets are unevenly distributed across categories. The training data consists of 310,355 documents distributed unevenly among the categories (with a minimum of 1,747 and maximum of 83,668 documents per category), and the testing set contains 10,000 documents unevenly distributed among categories (with a minimum of 56 and maximum of 2,696 documents per category).

### 4.2 Measuring classification performance

In the Chinese experiments, where 6 binary classification problems are formulated, we measured classification performance by *micro-averaged* F-measure scores. To calculate the micro-averaged score, we formed an aggregate confusion matrix by adding up the individual confusion matrices from each category. The micro-averaged precision, recall, and F-measure can then be computed based on the aggregated confusion matrix.

For the Japanese experiments, we measured *overall accuracy* and the *macro-averaged F-measure*. Here the precision, recall, and F-measures of each individual category can be computed based on a $|C| \times |C|$ confusion matrix. Macro-averaged scores can be computed by averaging the individual scores. The *overall accuracy* is computed by dividing the number of correctly identified documents (summing the numbers across the diagonal) by the total number of test documents.

### 4.3 Results on Chinese data

Table 1 gives the results of the character level language modeling approach, where rows correspond to different smoothing techniques. Columns correspond to different $n$-gram order $n = 1, 2, 3, 4$. The entries are the micro-average F-measure. (Note that the naive Bayes result corresponds to $n$-gram order 1 with *add one* smoothing, which is italicized in the table.) The results the ad hoc OOP classifier, and for the SVM classifier are shown in Table 2 and Table 3 respectively, where the columns labeled "Feature #" are the number of features selected.

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Add-one | *0.856* | 0.802 | 0.797 | 0.805 |
| Absolute | 0.856 | 0.868 | 0.867 | 0.868 |
| Good-Turing | 0.856 | 0.863 | 0.861 | 0.862 |
| Linear | 0.857 | 0.861 | 0.861 | 0.865 |
| Witten-Bell | 0.857 | 0.860 | 0.865 | 0.864 |

Table 1: Results of character level language modeling classifier on Chinese data.

| Feature # | Micro-F1 | Feature # | Micro-F1 |
|---|---|---|---|
| 100 | 0.7808 | 500 | 0.7848 |
| 200 | 0.8012 | 1000 | 0.7883 |
| 300 | 0.8087 | 1500 | 0.7664 |
| 400 | 0.7889 | 2000 | 0.7290 |

Table 2: Results of the character level OOP classifier on Chinese data.

| Feature # | Micro-F1 | Feature # | Micro-F1 |
|---|---|---|---|
| 100 | 0.811 | 500 | 0.817 |
| 200 | 0.813 | 1000 | 0.817 |
| 300 | 0.817 | 1500 | 0.815 |
| 400 | 0.816 | 2000 | 0.816 |

Table 3: Results of the character level SVM classifier on Chinese data.

### 4.4 Results on Japanese data

For the Japanese data, we experimented with byte level models (where in fact each Japanese character is represented by two bytes). We used byte level models to avoid possible character level segmentation errors that might be introduced, because we lacked the knowledge to detect misalignment errors in Japanese characters. The results of byte level language modeling classifiers on the Japanese data are shown in Table 4. (Note that the naive Bayes result corresponds to $n$-gram order 2 with *add one* smoothing, which is italicized in the table.) The results for the OOP classifier are shown in Table 5. Note that SVM is not applied in this situation since we are conducting multiple category classification directly while SVM is designed for binary classification. However, Aizawa (Aizawa, 2001) reported a performance of abut 85% with SVMs by converting the problem into a 24 binary classification problem and by performing word segmentation as preprocessing.

| Feature # | Accuracy | Macro-F |
|---|---|---|
| 100 | 0.2044 | 0.1692 |
| 200 | 0.2830 | 0.2308 |
| 300 | 0.3100 | 0.2677 |
| 400 | 0.3616 | 0.3118 |
| 500 | 0.3682 | 0.3295 |
| 1000 | 0.4416 | 0.4073 |
| 2000 | 0.4990 | 0.4510 |
| 3000 | 0.4770 | 0.4315 |
| 4000 | 0.4462 | 0.3820 |
| 5000 | 0.3706 | 0.3139 |

Table 5: Results of byte level OOP classifier on Japanese data.

## 5 Discussion and analysis

We now give a detailed analysis and discussion based on the above results. We first compare the language model based classifiers with other classifiers, and then analyze the influence of the order $n$ of the $n$-gram model, the influence of the smoothing method, and the influence of feature selection in tradition approaches.

### 5.1 Comparing classifier performance

Table 6 summarizes the best results obtained by each classifier. The results for the language model (LM) classifiers are better than (or at least comparable to ) other approaches for both the Chinese and Japanese data, while avoiding word segmentation. The SVM result on Japanese data is obtained from (Aizawa, 2001) where word segmentation was performed as a preprocessing. Note that SVM classifiers do not perform as well in our Chinese text classification as they did in English text classification (Dumais, 1998), neither did they in Japanese text classification (Aizawa, 2001). The reason worths further investigations.

Overall, the language modeling approach appears to demonstrate state of the art performance for Chinese and Japanese text classification. The reasons for the improvement appear to be three-fold: First, the language modeling approach always considers every feature during classification, and can thereby avoid an error-prone feature selection process. Second, the use of $n$-grams in the model relaxes the restrictive independence assumption of naive Bayes. Third, the techniques of statistical language modeling offer better smoothing methods for coping with features that are unobserved during training.

| LM | NB | OOP | SVM |
|---|---|---|---|
| Chinese Character Level | | | |
| 0.868 | 0.856 | 0.8087 | 0.817 |
| Japanese Byte Level | | | |
| 0.84 | 0.66 | 0.4990 | 85% (Aizawa, 2001) |

Table 6: Comparison of best classifier results

### 5.2 Influence of the $n$-gram order

The order $n$ is a key factor in $n$-gram language modeling. An order $n$ that is too small will not capture sufficient information to accurately model character dependencies. On the other hand, a context $n$ that is too large will create sparse data problems in training. In our Chinese experiments, we did not observe significant improvement when using higher order $n$-gram models. The reason is due to the early onset of sparse data problems. At the moment, we only have limited training data for Chinese data set (1M in size, 500 documents per class for training). If more training data were available, the higher order models may begin to show an advantage. For example, in the larger Japanese data set (average 7M size, 12,931 documents per class for training) we

| $n$ | Add-one | | Absolute | | Good-Turing | | Linear | | Witten-Bell | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Accu. | F-Mac | Accu. | F-Mac | Accu. | F-Mac | Accu. | F-Mac | Accu. | F-Mac |
| 1 | 0.33 | 0.29 | 0.33 | 0.29 | 0.34 | 0.29 | 0.34 | 0.29 | 0.34 | 0.29 |
| 2 | *0.66* | 0.63 | 0.66 | 0.62 | 0.66 | 0.61 | 0.66 | 0.63 | 0.66 | 0.62 |
| 3 | 0.77 | 0.68 | 0.75 | 0.72 | 0.75 | 0.72 | 0.76 | 0.73 | 0.75 | 0.72 |
| 4 | 0.74 | 0.51 | 0.81 | 0.77 | 0.81 | 0.76 | 0.82 | 0.76 | 0.81 | 0.77 |
| 5 | 0.69 | 0.42 | 0.83 | 0.77 | 0.83 | 0.76 | 0.83 | 0.76 | 0.83 | 0.77 |
| 6 | 0.66 | 0.42 | 0.84 | 0.76 | 0.83 | 0.75 | 0.83 | 0.75 | 0.84 | 0.77 |
| 7 | 0.64 | 0.38 | 0.84 | 0.75 | 0.83 | 0.74 | 0.83 | 0.74 | 0.84 | 0.76 |
| 8 | 0.62 | 0.31 | 0.83 | 0.74 | 0.83 | 0.73 | 0.83 | 0.73 | 0.84 | 0.76 |

Table 4: Results of byte level language model classifier on Japanese data.

observe an obvious increase in classification performance with higher order models (Table 4). However, here too, when $n$ becomes too large, overfitting will begin to occur, as better illustrated in Figure 1.

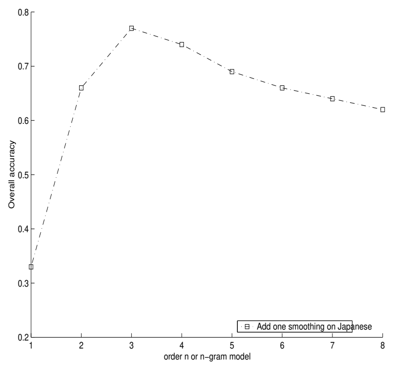


Figure 1: Effects of order of $n$-gram language models

### 5.3 Influence of smoothing techniques

Smoothing plays an key role in language modeling. Its effect on classification is illustrated in Figure 2. In both cases we have examined, *add one* smoothing is obviously the worst smoothing technique, since it systematically overfits much earlier than the more sophisticated smoothing techniques. The other smoothing techniques do not demonstrate a significant difference in classification accuracy on our Chinese and Japanese data, although they do show a difference in the perplexity of the language models themselves (not shown here to save space). Since our goal is to make a final decision based on the *ranking* of perplexities, not just their absolute values, a superior smoothing method in the sense of perplexity reduction does not necessarily lead to a better decision from the perspective of categorization accuracy.

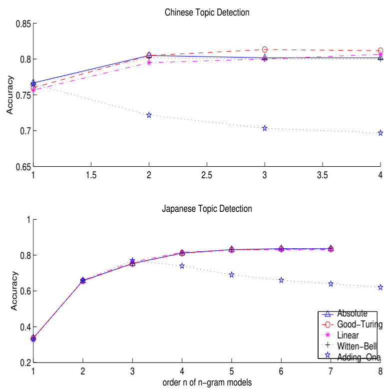


Figure 2: Effects of the smoothing techniques

### 5.4 Influence of feature selection

The number of features selected is a key factor in determining the classification performance of the OOP and SVM classifiers, as shown in Figure 3. Obviously the OOP classifier is adversely affected by increasing the number of selected features. By contrast, the SVM classifier is very robust with respect to the number of features, which is expected because the complexity of the SVM classifier is determined by the number of support vectors, not the dimensionality of the feature space. In practice, some heuristic search methods are normally used to obtain an optimal subset of features. However, in our language modeling based approach, we avoid explicit feature selection by considering all possible features and the importance of each individual feature is measured by its contribution to the perplexity (or entropy) value.

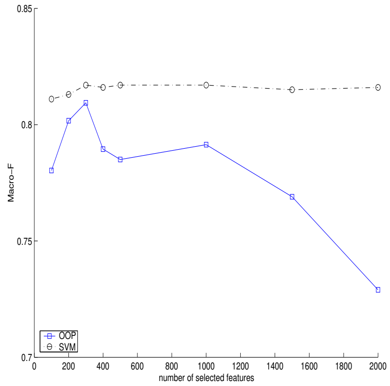


Figure 3: Effects of the number of selected features

### 5.5 Related Work

The use of $n$-gram models has also been extensively investigated in information retrieval. However, unlike previous research (Cavnar and Trenkle, 1994; Damashek, 1995), where researchers have used $n$-grams as features for a traditional feature selection process and then deployed classifiers based on calculating feature-vector similarities, we consider *all* $n$-grams as features and determine their importance implicitly by assessing their contribution to perplexity. In this way, we avoid an error prone feature selection step.

Language modeling for text classification is a relatively new area. In principle, any language model can be used to perform text categorization. However, $n$-gram models are extremely simple and have been found to be effective in many applications. Teahan and Harper (Teahan and Harper, 2001) used a PPM (prediction by partial matching) model for text categorization where they seek a model that obtains the best compression on a new document.

## 6 Conclusion

We have presented a simple language model based approach without word segmentation for Chinese and Japanese text classification. By comparison to three standard text classifiers, the language modeling approach consistently demonstrates better classification accuracies while avoiding word segmentation and feature selection. Although straightforward, the language modeling approach appears to give state of the art results for Chinese and Japanese text classification.

It has been found that word segmentation in Chinese text retrieval is tricky and the relationship between word segmentation and retrieval performance is not monotonic (Peng et al., 2002). However, since text classification and text retrieval are two different tasks, it is not clear whether the same relationship exists in text classification context. We are currently investigating this issue and interesting findings have already been observed.

## References


A. Aizawa. 2001. Linguistic Techniques to Improve the Performance of Automatic Text Categorization. *Proceedings NLPRS2001.*

W. Cavnar and J. Trenkle. 1994. N-Gram-Based Text Categorization. *Proceedings of SDAIR-94*

S. Chen and J. Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. *TR-10-98, Harvard University*

M. Damashek. 1995. Gauging Similarity with N-Grams: Language-Independent Categorization of Text? *Science*, 267(10), pages 843-848.

S. Dumais, J. Platt, D. Heckerman, and M. Sahami 1998. Inductive Learning Algorithms and Representations for Text Categorization. *Proceedings of CIKM98*

J. He, A. Tan, and C. Tan. 2001. On Machine Learning Methods for Chinese Documents Classification. *Applied Intelligence's Special Issue on Text and Web Mining*

T. Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of the ECML-1998*

F. Peng, X. Huang, D. Schuurmans, and N. Cercone. 2002. Investigating the Relationship of Word Segmentation Performance and Retrieval Performance in Chinese IR. *Proceedings of COLING2002*

F. Peng and D. Schuurmans. 2003. Combining Naive Bayes and N-Gram Language Models for Text Classification. *Proceedings of ECIR2003*

F. Sebastiani. 2002. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1).

W. Teahan and D. Harper. 2001. Using Compression-Based Language Models for Text Categorization. *Proceedings of LMIR2001*

V. Vapnik. 1995. *The Nature of Statistical Learning Theory.* Springer-Verlag.

Y. Yang. 1999. An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval Journal*, 1/2.