Augmenting Naive Bayes Classifiers with Statistical Language Models *

Fuchun Peng

Center for Intelligent Information Retrieval,

Department of Computer Science, University of Massachusetts at Amherst

140 Governors Drive, Amherst, MA, USA, 01003

fuchun@cs.umass.edu

Dale Schuurmans

Department of Computing Science, University of Alberta

Edmonton, Alberta, Canada, T6G 2E8

dale@cs.ualberta.ca

Shaojun Wang

Department of Computing Science, University of Alberta

Edmonton, Alberta, Canada, T6G 2E8

swang@cs.ualberta.ca

Abstract

We augment naive Bayes models with statistical n-gram language models to address shortcomings of the standard naive Bayes text classifier. The result is a generalized naive Bayes

*Most research was conducted while the authors were at the School of Computer Science at University of Waterloo, Canada.

classifier which allows for a local Markov dependence among observations; a model we refer to as the **C***hain* **A***ugmented* **N***aive Bayes* (CAN) Bayes classifier. CAN models have two advantages over standard naive Bayes classifiers. First, they relax some of the independence assumptions of naive Bayes—allowing a local Markov chain dependence in the observed variables—while still permitting efficient inference and learning. Second, they permit straightforward application of sophisticated smoothing techniques from statistical language modeling, which allows one to obtain better parameter estimates than the standard Laplace smoothing used in naive Bayes classification. In this paper, we introduce CAN models and apply them to various text classification problems. To demonstrate the language independent and task independent nature of these classifiers, we present experimental results on several text classification problems—authorship attribution, text genre classification, and topic detection—in several languages—Greek, English, Japanese and Chinese. We then systematically study the key factors in the CAN model that can influence the classification performance, and analyze the strengths and weaknesses of the model.

Index Keywords:

naive Bayes, text classification, n-gram language models, smoothing

1 Introduction

Text classification concerns the problem of automatically assigning given text passages (paragraphs or documents) into predefined categories. Due to the rapid explosion of texts in digital form, text classification has become an important area of research, owing to the need to automatically organize and index large text collections in various ways. Such techniques are currently being applied in many areas, including spam detection, language identification [43], authorship attribution [46], text genre classification [26, 46], topic identification [11, 32, 35, 54], and subjective sentiment classification [38, 50].

Many standard machine learning techniques have been applied to automated text categorization problems, such as naive Bayes classifiers, support vector machines, linear least square models, neural networks, and K-nearest neighbor classifiers [45, 54]. Naive Bayes classifiers have been proven successful in text classification [33, 35, 42], despite the simplicity of the model and the restrictiveness of the independence assumptions it makes. Domingos and Pazzanni [9] observe that naive Bayes classifiers can obtain near optimal misclassification error even when the independence assumption is strongly violated. Nevertheless, it is commonly thought that relaxing the independence assumption of naive Bayes ought to allow for superior text classification [33], and it has been shown in practice that functionally dependent attributes can indeed improve classification accuracy in some cases [14, 41].

A significant amount of research has been conducted on relaxing the naive Bayes independence assumption in machine learning research. A popular extension is the **T***ree* **A***ugmented* **N***aive* Bayes classifier (TAN) [14] which allows for a tree-structured dependence among observed variables in addition to the traditional dependence on the hidden "root" variable. However, learning tree structured Bayesian network is not trivial [25, 55], and this model has rarely been used in text classification applications. In this paper we investigate a convenient alternative that lies between pure naive Bayes and TAN Bayes models in the strength of its assumptions; namely, the **C***hain* **A***ugmented* **N***aive Bayes* (CAN) model. A CAN Bayes model simplifies the TAN model by restricting dependencies among observed variables to form a Markov chain instead of a tree. Interestingly, it turns out that the model that results is closely related to the n-gram language models that have been widely studied in statistical natural language modeling and speech recognition [13, 27]. We augment the naive Bayes text classifier by including observation dependencies that form a Markov chain, and use techniques from statistical n-gram language modeling to learn and apply these models. The result is a combination of naive Bayes and statistical n-gram methods that yields simple yet surprisingly effective classifiers.

An advantage of using the CAN model, which naturally incorporates the use of statistical language models for classification, is that they also allow the straightforward use of advanced smoothing techniques from statistical language modeling to obtain better parameter estimates. In the standard naive Bayes approach, Laplace smoothing is commonly used to avoid zero probability estimates. However, Laplace smoothing is usually less effective than many other smoothing techniques from statistical language modeling. We consider four more advanced smoothing techniques that have been commonly used in n-gram language modeling to improve naive Bayes classifier, and achieve significant improvements.

In addition to improving the standard naive Bayes model, CAN models also have advantages over other traditional text classifiers. A common aspect of traditional text classification approaches is that they treat text categorization as a standard classification problem, and thereby reduce the learning process to two simple steps: feature engineering, and discriminant learning over the feature space. Of these two steps, feature engineering is critical to achieving good performance in practice. Once good features are identified, almost any reasonable technique for learning a classifier seems to perform well [44]. However, relying upon an explicit feature selection process has several drawbacks. First, feature construction is usually language and task dependent. Various techniques such as stop-word removal or stemming require language specific knowledge to design adequately. Moreover, most traditional text classifiers work on word level features, whereas identifying words from character sequences is hard in many Asian languages—such as Chinese or Japanese—and any word-based approach must suffer added complexity in coping with segmentation errors. Second, feature selection is task dependent. For example, tasks like authorship attribution or genre classification require attention to linguistic style markers [46], whereas topic detection systems rely more heavily on bag of words features. Third, there are an enormous number of possible features to consider in text classification problems, and standard feature selection approaches do not always cope well in such circumstances. For example, given an enormous 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. Consequently, throwing away uncommon features is usually not an appropriate strategy in this domain [1]. A better approach is to use as many features as possible.

CAN models provide a convenient and effective way to mitigate these difficulties. First, they deal with the feature explosion problem by employing simple back-off estimators from statistical language modeling, which avoids explicit feature selection by considering all n-grams as features and measures their importance implicitly by their contribution to language models. Thus, CAN models avoid the need for explicit feature selection, which is a critical but error prone process. Second, *character level* CAN models can be deployed in order to avoid the word segmentation issues that arise in Asian languages, and thus provide a simple avenue toward achieving language independent and task independent text classification.

The remainder of the paper is organized as follows. First in Section 2 we describe the naive Bayes classifier and then present the basic n-gram language model in Section 3. These two models form the basis of the chain augmented naive Bayes classifier we propose in Section 4. We then experimentally evaluate the proposed classifier on various text classification problems in various languages in Section 5. A detailed discussion and analysis is presented in Section 6. Finally, we discuss related work in Section 7 and conclude in Section 8.

2 The naive Bayes text classifier

Text classification is the problem of assigning a document D to one of a set of $|C|$ pre-defined categories $C = \{c_1, c_2, ..., c_{|C|}\}$. Normally a supervised learning framework is used to train a text classifier, where a learning algorithm is provided a set of N labeled training examples $\{(d_i, c_i) : i = 1, ..., N\}$ from which it must produce a classification function $F : D \rightarrow C$ that maps documents to categories. Here d_i denotes the ith training document and c_i is the corresponding category label of d_i. We use the random variables D and C to denote the document and category values respectively. A popular learning algorithm for text classification is based on a simple application of *Bayes' rule* [10, chapter 10]:[1]

$$P(C = c|D = d) = \frac{P(C = c) \times P(D = d|C = c)}{P(D = d)} \tag{1}$$

where d and c are instances of D and C, $P(D = d) = \sum_{c \in C} P(C = c)P(D = d|C = c)$. To simplify the presentation, we re-write Equ. (1) as

$$P(c|d) = \frac{P(c) \times P(d|c)}{P(d)} \tag{2}$$

[1]Other popular learning algorithms for text classification include decision tree learning, support vector machines, regression methods, neural network, rule learning methods, and on-line learning [45].

Bayes' rule decomposes the computation of a posterior probability into the computation of a likelihood and a prior probability. In text classification, a document d is normally represented by a vector of K attributes $d = (v_1, v_2, ...v_K)$.[2] Computing $p(d|c)$ in this case is not generally trivial, since the space of possible documents $d = (v_1, v_2, ...v_K)$ is vast. To simplify this computation, the naive Bayes model introduces an additional assumption that all of the attribute values, v_j, are independent given the category label, c. That is, for $i \neq j$, v_i and v_j are conditionally independent given c. This assumption greatly simplifies the computation by reducing Equ. (2) to

$$P(c|d) \quad = \quad P(c) \times \frac{\prod_{j=1}^{K} P(v_j|c)}{P(d)} \tag{3}$$

Based on Equ. (3), maximum a posterior (MAP) classifier can be constructed by seeking the optimal category which maximizes the posterior $P(c|d)$:

$$c^* = \arg\max_{c \in C} \ \{P(c|d)\} \tag{4}$$

$$= \arg\max_{c \in C} \left\{ P(c) \times \frac{\prod_{j=1}^{K} P(v_j|c)}{P(d)} \right\} \tag{5}$$

$$= \arg\max_{c \in C} \left\{ P(c) \times \prod_{j=1}^{K} P(v_j|c) \right\} \tag{6}$$

Note that the step from Equ. (5) to Equ. (6) is valid because $P(d)$ is a constant for every category c. A MAP classifier (Equ. (4)) is optimal in the sense of minimizing *zero-one* loss (misclassification error) [10]. If the independence assumption holds, then a classifier based on Equ. (6) is also optimal [10]. The prior distribution $P(c)$ can be used to incorporate additional assumptions about the relative frequencies of classes. Two commonly used prior distributions are the uniform distribution and the empirical distribution estimated with sample frequencies.

There are several variants of naive Bayes classifiers, including the binary independence model, the multinomial model, the Poisson model, and the negative binary independence model [12]. It has been shown that for text classification applications, the multinomial model is most often the best choice [12, 35], therefore we will only consider the multinomial naive Bayes model in this paper.

[2] Attributes are also called features in many papers.

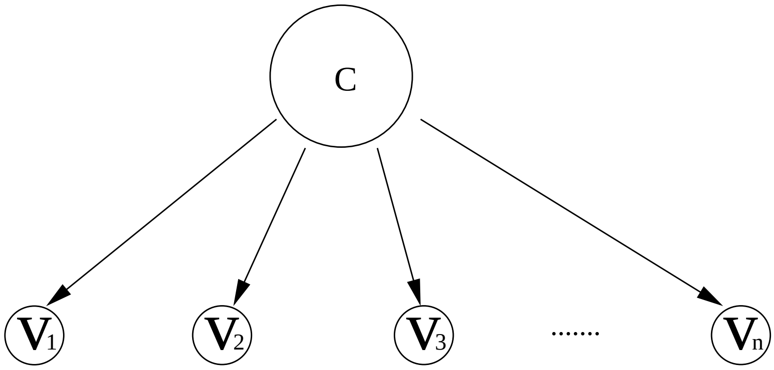

Figure 1: Graphical model of a naive Bayes classifier where each leaf node is independent of each other given its parent.

Figure 1 gives a graphical representation of the multinomial naive Bayes model, showing that each attribute node is independent of the other attributes given the class label C.

The parameters of a multinomial naive Bayes classifier are given by $\Theta = \{\theta_j^c = P(v_j|c) : j = 1, ..., K; c = 1, ..., |C|\}$. The likelihood of a given set of documents D^c for a given category c is given by

$$P(D^c|\Theta) \;=\; \frac{N^c!}{\prod_j N_j^c!} \prod_j (\theta_j^c)^{N_j^c} \tag{7}$$

where N_j^c is the frequency of attribute j occurring in D^c and $N^c = \sum_j N_j^c$. A maximum likelihood estimate yields the parameter estimates

$$\theta_j^c \;=\; \frac{N_j^c}{N^c} \tag{8}$$

Note that Equ. (8) puts zero probability on attribute values that do not actually occur in D^c (i.e., $N_j^c = 0$). Unfortunately, a zero estimate can create significant problems when we classify a new document, for example, when we encounter a new attribute value that has not been observed in the training corpus D^c. To overcome this problem, Laplace smoothing is usually used to avoid zero probability estimates in practice:

$$\theta_j^c \;=\; \frac{N_j^c + a_j}{N^c + a} \tag{9}$$

where a_j is some constant and $a = \sum_j a_j$. A special case of Laplace smoothing is *add one* smoothing [34, chapter 6] obtained by setting $a_j = 1$. However, Laplace smoothing is not as effective in language modeling as some other smoothing techniques [5, 22]. We will show that more advanced smoothing techniques can be used to improve naive Bayes classifiers, and therefore play an important role in developing effective text classifiers using naive Bayes models.

We now describe the next main component of our models—statistical n-gram language models—which we will use to later augment naive Bayes classifiers.

3 Statistical n-gram language modeling

Although the dominant motivation for language modeling has come from speech recognition, statistical language models have recently become more widely used in many other application areas, including information retrieval [18, 39].

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

$$\mathit{Perplexity} = \sqrt[T]{\frac{1}{P(w_1 ... w_T)}} \tag{10}$$

$$\mathit{Entropy} = \log_2 \mathit{Perplexity} \tag{11}$$

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

$$P(w_1 w_2 ... w_T) = \prod_{i=1}^{T} P(w_i | w_1 ... w_{i-1}) \tag{12}$$

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

$$P(w_i|w_{i-n+1}...w_{i-1}) \quad = \quad \frac{\#(w_{i-n+1}...w_i)}{\#(w_{i-n+1}...w_{i-1})} \tag{13}$$

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 linear interpolation or back-off estimator. Linear interpolation models involves an EM procedure to optimize the weight for each component [5, 22] and back-off model is relatively simple. To keep the simplicity of naive Bayes model, we use back-off models.

$$P(w_i|w_{i-n+1}...w_{i-1}) = \begin{cases} \hat{P}(w_i|w_{i-n+1}...w_{i-1}), \\ \qquad \text{if } \#(w_{i-n+1}...w_i) > 0 \\ \beta(w_{i-n+1}...w_{i-1}) \times P(w_i|w_{i-n+2}...w_{i-1}), \\ \qquad \text{otherwise} \end{cases} \tag{14}$$

where

$$\hat{P}(w_i|w_{i-n+1}...w_{i-1}) = \frac{\text{discount}\, \#(w_{i-n+1}...w_i)}{\#(w_{i-n+1}...w_{i-1})} \tag{15}$$

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

$$\beta(w_{i-n+1}...w_{i-1}) = \frac{1 - \sum_{x:\#(w_{i-n+1}...w_{i-1}x)>0} \hat{P}(x|w_{i-n+1}...w_{i-1})}{1 - \sum_{x:\#(w_{i-n+1}...w_{i-1}x)>0} \hat{P}(x|w_{i-n+2}...w_{i-1})} \tag{16}$$

An n-gram is first matched against the language model to see if it has been observed in the training corpus. If that fails, the n-gram is then reduced to an $n-1$-gram by shortening the context by one word. The discounted probability (15) can then be computed using different smoothing approaches. The standard smoothing approaches we consider in this paper include linear smoothing [37, 52], absolute smoothing[37], Good-Turing smoothing [24] and Witten-Bell smoothing [52].

To describe the smoothing techniques, let n_i denote the number of events which occur *exactly* i times in training data [37].

Absolute smoothing Here the frequency of a word is subtracted by a constant b so that the discounted probability (Equ. 15) is calculated as

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

where b is often defined as $b = \frac{n_1}{n_1+2n_2}$.

Linear smoothing Here the discounted probability is calculated as

$$\hat{P}(w_i|w_{i-n+1}...w_{i-1}) = \left(1 - \frac{n_1}{T}\right) \times \frac{\#(w_{i-n+1}...w_i)}{\#(w_{i-n+1}...w_{i-1})}$$

where T is the number of uni-grams, which corresponds to the number of words in the training data [37].

Good-Turing smoothing Good-Turing smoothing discounts the frequency of r by $GT_r = (r+1)\frac{n_{r+1}}{n_r}$ and the discounted probability is calculated as [24]:

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

There are different variants of this standard Good-Turing smoothing. These include Katz's variant [24] and Church & Gale's variant [6]. We implemented Katz's variant.

Witten-Bell smoothing Witten-Bell smoothing is very similar to Laplace smoothing, except it reserves probability mass for out of vocabulary (OOV) values, whereas Laplace smoothing does not.[3] Here the discounted probability is calculated as

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

where $D(w_{i-n+1}...w_{i-1})$ is the number of distinct words that can follow $w_{i-n+1}...w_{i-1}$ in the training data [52]. In the uni-gram model, this corresponds to the size of vocabulary.

We now show how the naive Bayes and statistical n-gram language models can be combined to form the chain augmented naive Bayes classifier.

4 Applying n-gram 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. An n-gram language model can be applied to text classification in a similar manner to a naive Bayes model. That is, we categorize a document according to

$$c^* = \arg\max_{c \in C}\{P(c|d)\} \tag{17}$$

[3]Witten-Bell smoothing is a misnomer since it was actually invented by Alistair Moffat [36], and is called method C in PPM text compression. We thank William Teahan for pointing this out (personal communication).

Using Bayes rule, this can be rewritten as

$$\begin{aligned} c^* &= \arg\max_{c \in C} \{P(c)P(d|c)\} && (18) \\ &= \arg\max_{c \in C} \left\{ P(c) \prod_{i=1}^{T} P(w_i|w_{i-n+1}...w_{i-1}, c) \right\} && (19) \\ &= \arg\max_{c \in C} \left\{ P(c) \prod_{i=1}^{T} P_c(w_i|w_{i-n+1}...w_{i-1}) \right\} && (20) \end{aligned}$$

Here, $P(d|c)$ is the likelihood of d under category c, which can be computed by an n-gram language model. Likelihood is related to perplexity and entropy by Equ. (10) and Equ. (11). The prior $P(c)$ can be estimated from training data or can be used to incorporate more assumptions, similar to the standard naive Bayes model. In our case, $P_c(w_i|w_{i-n+1}...w_{i-1})$ is computed using a back-off model introduced in previous section.

Thus our approach overall 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. (20).

Note that n-gram classifiers are in fact a straightforward generalization of naive Bayes: a unigram classifier with Laplace smoothing corresponds exactly to the traditional naive Bayes classifier. However, n-gram language models, for larger n, possess many advantages over naive Bayes classifiers, including modeling longer context and exploiting superior smoothing techniques in the presence of sparse data. For naive Bayes text classifiers, attributes (words) are considered independent of each other given the category. However, in a language modeling based approach, this is enhanced by considering a Markov dependence between adjacent words. Due to this similarity, we refer the n-gram augmented naive Bayes classifier as a **C***hain* **A***ugmented* **N***aive Bayes* classifier (CAN).

Another notable advantage of the language modeling based approach is that it does not incorporate an explicit feature selection procedure. That is, instead, it considers all possible n-grams as features. Their importance is implicitly considered by their contribution to the quality of language modeling (in sense of perplexity in Equ. (10)). The back-off smoothing technique effectively deals with feature explosion. The over-fitting problems associated with the subsequent feature explosion are nicely handled by applying the backoff smoothing techniques outlined above.

Note that the basic unit used in the language models described above is the word. However, we can also consider text as a concatenated sequence of *characters* instead of words. The formulation of a character based language model is the same as above, except that the size of the vocabulary is much smaller, which greatly reduces the sparse data problem. Character level models also avoid the word segmentation problems that occur in many Asian languages such as Chinese and Japanese, and thus allows constructing language independent classifiers. We also explore this alternative below.

5 Experimental comparison

To systematically evaluate the CAN model, we investigate several different text classification tasks on various languages. Specifically, we consider Greek authorship attribution, Greek genre classification, and topic detection in English, Japanese, and Chinese.

5.1 Classification performance and experimental paradigm

In many cases, classification can be carried out directly based on Equ. (17). However some classifiers, such as SVMs, are specifically designed only for binary classification problems. For these classifiers, given a $|C|$ category classification problem, we have to convert it to a set of binary classification problems.

Different performance measures can be used to assess these two different unique classification approaches. For the sake of consistency with previous research, we experiment with both approaches in different data. In the Chinese topic detection experiments, there are 6 classes, so we formulate 6 binary classification problems. In Reuters topic detection experiments, there are 10 classes, so we formulate 10 binary classification problems. In both of these cases, we measure classification performance by *micro-averaged F-measure*. 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.

In other experiments, we measured *overall accuracy* and *macro-averaged F-measure*. (Micro-averaged values are not available in these cases.) 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. In most experiments, we use numeric precision to 4 digits. However in some experiments (Greek authorship attribution, Japanese topic detection, Chinese topic detection), we can not obtain higher numeric precision due to the size of the used test sets.

In all our experiments, we first report results of character level CAN models which consider text as a sequence of characters. However, for western languages such as English and Greek, we also report results of using *word* level CAN models which consider text as a sequence of words.

5.2 Authorship attribution

The first text categorization problem we examined was author attribution. A famous example is the case of the *Federalist Papers*, of which twelve instances are claimed to have been written both by Alexander Hamilton and James Madison [19]. We considered a data set used by Stamatatos et al. [46] consisting of 20 texts written by 10 different modern Greek authors (totaling 200 documents). In each case, 10 texts from each author were used for training and the remaining 10 for testing (using the same split as [46]). The specific authors that appear are shown in Table 1.

The results of using character level models are shown in Table 2. In our experiments, we obtained the best performance by using a tri-gram model with absolute smoothing. The best accuracy we obtained is **90%**. This compares favorably to the best accuracy reported in [46] of 72%.[4] The 18% accuracy improvement is surprising given the relative simplicity of our method.

As a typical western language, Greek words are separated by white space, so we can also apply word level models to this data. Table 3 shows the results of the word level model. Here the perfor-

[4]Note that Stamatatos et al.'s measures of *identification error* and *average error* correspond to our *recall* and *overall accuracy* measures respectively.

Code	Author Name	Train size (characters)
B0	S. Alaxiotis	77295
B1	G. Babiniotis	75965
B2	G. Dertilis	66810
B3	C. Kiosse	102204
B4	A. Liakos	89519
B5	D. Maronitis	36665
B6	M. Ploritis	72469
B7	T. Tasios	80267
B8	K. Tsoukalas	104065
B9	G. Vokos	64479

Table 1: Authors in the Greek authorship attribution data set

n	Absolute		Laplace		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	0.57	0.53	0.56	0.51	0.55	0.49	0.55	0.49	0.55	0.49
2	0.85	0.84	0.82	0.79	0.80	0.75	0.84	0.83	0.84	0.82
3	**0.90**	0.89	0.70	0.64	0.79	0.72	0.89	0.88	0.89	0.87
4	0.87	0.85	0.47	0.39	0.79	0.72	0.85	0.82	0.88	0.86
5	0.86	0.85	0.35	0.29	0.79	0.72	0.87	0.85	0.86	0.83
6	0.86	0.83	0.16	0.10	0.79	0.73	0.87	0.85	0.86	0.83

Table 2: Results on Greek authorship attribution using character level models

mance increases to 96%, which is better than character level models. However, such a significant improvement is not always observed in other experiments, as we will see below.

n	Absolute		Laplace		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	**0.96**	0.95	0.76	0.72	0.93	0.92	**0.96**	0.95	0.61	0.57
2	**0.96**	0.95	0.60	0.56	0.95	0.94	0.95	0.94	0.83	0.78
3	**0.96**	0.95	0.54	0.51	0.95	0.94	0.95	0.94	0.83	0.78
4	**0.96**	0.95	0.56	0.53	0.95	0.94	0.95	0.94	0.84	0.80

Table 3: Results on Greek authorship attribution using word level models

5.3 Text genre classification

The second problem we examined was text genre classification, which is an important application in natural language processing [26] and information retrieval [31]. We considered a Greek data set used by [46] consisting of 20 texts of 10 different styles extracted from various sources (200 documents in total). For each style, we used 10 texts as training data and the remaining 10 as testing data (using the same split as in [46]). The 10 different text genres and their original sources are shown in Table 4.

code	Genre	Source
0	Press editorial	Newspaper TO BHMA
1	Press reportage	Newspaper TO BHMA
2	Academic prose	Journal of archives of hellonic pathology
3	Official documents	High court decisions, ministerial decisions
4	Literature	Various pages
5	Recipes	Magazine NETLIFE
6	Curriculum Vitae	Various pages
7	Interviews	Newspaper TO BHMA
8	Planned speeches	Ministry of defense
9	Broadcast news, scripted	Radio station, FLASH 9.61

Table 4: 10 different text genres

The results of the character level model are shown in Table 5. The **86%** accuracy obtained with bi-gram models compares favorably to the 82% reported in [46], which again is based on a much deeper NLP analysis.

n	Absolute		Laplace		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	0.31	0.55	0.30	0.54	0.30	0.54	0.30	0.54	0.30	0.54
2	**0.86**	0.86	0.72	0.69	0.60	0.52	0.82	0.81	**0.86**	0.86
3	0.77	0.75	0.50	0.41	0.65	0.59	0.79	0.77	0.85	0.85
4	0.69	0.65	0.38	0.29	0.58	0.50	0.74	0.69	0.76	0.74
5	0.66	0.61	0.38	0.29	0.56	0.49	0.69	0.66	0.73	0.70
6	0.62	0.57	0.38	0.29	0.49	0.53	0.67	0.63	0.71	0.68
7	0.63	0.58	0.33	0.21	0.49	0.53	0.66	0.62	0.70	0.68

Table 5: Results on Greek text genre classification using character level models

As before, we can again apply word level models to Greek text genre classification. Here we obtain 81% accuracy using absolute smoothing and uni-grams ($n = 1$). The standard naive Bayes model (n=1 and Laplace smoothing) achieves only 56% performance in this case. Here word level models perform worse than character level models.

n	Absolute		Laplace		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	0.81	0.78	0.56	0.48	0.76	0.73	0.77	0.76	0.50	0.42
2	0.80	0.77	0.59	0.61	0.79	0.76	0.77	0.74	0.61	0.56
3	0.80	0.77	0.61	0.62	0.78	0.75	0.77	0.74	0.60	0.56

Table 6: Results on Greek text genre classification using word level models

5.4 Topic detection

The third problem we examined was topic detection in text, which is a heavily researched text categorization problem [11, 32, 35, 45, 54]. Here we demonstrate the language independence of the character based language modeling approach by considering experiments on English, Japanese and Chinese data.

5.4.1 English 20 newsgroup data

The English 20 Newsgroup data was originally collected and used for text categorization by Lang [30] and has been widely used in topic detection research [35, 40].[5] This collection consists of 19,974 non-empty documents distributed evenly across 20 newsgroups. We use the newsgroups to form categories, and randomly select 80% of the documents to be used for training and the remaining 20% for testing. Table 7 shows the 20 categories appearing in the newsgroup data.

We first considered text to be a sequence of characters, and learned character level n-gram models. The resulting classification accuracies are reported in in Table 8. All results are averaged across 5 random runs. In the character model, with tri-gram (or higher order) models, we consistently obtain accurate performance, peaking at **89.23±0.33**% accuracy in the case of 5-gram models with

[5] http://www.ai.mit.edu/˜ jrennie/20Newsgroups/

Code	Newsgroup	Code	Newsgroup
0	alt.atheism	10	rec.sport.hockey
1	comp.graphics	11	sci.crypt
2	comp.os.ms-windows.misc	12	sci.electronics
3	comp.sys.ibm.pc.hardware	13	sci.med
4	comp.sys.mac.hardware	14	sci.space
5	comp.windows.x	15	soc.religion.christian
6	misc.forsale	16	talk.politics.guns
7	rec.autos	17	talk.politics.mideast
8	rec.motorcycles	18	talk.politics.misc
9	rec.sport.baseball	19	talk.religion.misc

Table 7: English 20 newsgroup data

Witten-Bell smoothing. These results compare favorably to the state of the art result of 87.5% accuracy reported in [40], which was based on a combination of SVMs with error correcting output coding (ECOC).

As before, we can also consider English text as a sequence of words and therefore also apply word models in this case. The results are shown in Table 9. Word level models were able to achieve 88.22±0.35% accuracy in this case. (Using the t-test, we find that the difference between the results of character level models and word level models are statistically significant to the $\alpha = 0.01$ significance level. The resulted p-value is 0.0027 and t-score is 4.2818.) Note that the word level model with Laplace smoothing and $n = 1$ is exactly equivalent to the traditional Naive Bayes classifier, which only achieves 84.93% accuracy in this case—consistent with previous findings [35].

n	Absolute		Laplace		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	0.2128	0.2034	0.2128	0.2033	0.2126	0.2032	0.2126	0.2032	0.2126	0.2032
2	0.6691	0.6505	0.6646	0.6451	0.6760	0.6569	0.6772	0.6584	0.6628	0.6437
3	0.8680	0.8635	0.8458	0.8385	0.8617	0.8571	0.8629	0.8596	0.8673	0.8621
4	0.8879	0.8849	0.8279	0.8197	0.8803	0.8775	0.8783	0.8760	0.8903	0.8870
5	0.8895	0.8867	0.7384	0.7267	0.8803	0.8764	0.8824	0.8806	**0.8923**	0.8896
6	0.8876	0.8852	0.6517	0.6453	0.8787	0.8759	0.8810	0.8794	0.8913	0.8883
7	0.8862	0.8831	0.5800	0.5759	0.8771	0.8742	0.8838	0.8821	0.8909	0.8875

Table 8: Topic detection results on English 20 newsgroup data using character models

n	Absolute		Laplace		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	0.8735	0.8694	*0.8493*	0.8412	0.8653	0.8612	0.8757	0.8717	0.8423	0.8343
2	0.8818	0.8781	0.4730	0.4718	0.8733	0.8686	**0.8822**	0.8790	0.8749	0.8711
3	0.8800	0.8766	0.3381	0.3416	0.8713	0.8668	0.8798	0.8769	0.8726	0.8693
4	0.8808	0.8774	0.3234	0.3274	0.8712	0.8667	0.8795	0.8766	0.8730	0.8699

Table 9: Topic detection results on English 20 newsgroup data using word models

5.4.2 Japanese data

Japanese topic detection is often thought to be more challenging than in English, because words are not white-space delimited in Japanese text. This fact seems to require word segmentation to be performed as a pre-processing step before further classification [1]. However, we avoid the need for explicit segmentation by simply using a character level (byte level) n-gram classifier.

We consider the Japanese topic detection data investigated by [1]. This data set was converted from the NTCIR-J1 data set originally created for Japanese text retrieval research. The data has 24 categories. The testing set contains 10,000 documents distributed unevenly between categories. We have obtained the experimental results shown in Table 10, which still show an **84%** accuracy rate on this problem (for 6-gram or higher order models). This is the same level of performance as that reported in [1], which uses an SVM approach with word segmentation, morphology analysis and feature selection.

n	Absolute		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	0.33	0.29	0.34	0.29	0.34	0.29	0.34	0.29
2	0.66	0.62	0.66	0.61	0.66	0.63	0.66	0.62
3	0.75	0.72	0.75	0.72	0.76	0.73	0.75	0.72
4	0.81	0.77	0.81	0.76	0.82	0.76	0.81	0.77
5	0.83	0.77	0.83	0.76	0.83	0.76	0.83	0.77
6	**0.84**	0.76	0.83	0.75	0.83	0.75	**0.84**	0.77
7	**0.84**	0.75	0.83	0.74	0.83	0.74	**0.84**	0.76
8	0.83	0.74	0.83	0.73	0.83	0.73	**0.84**	0.76

Table 10: Japanese topic detection results

5.4.3 Chinese data

Chinese poses the same word segmentation issues as Japanese. Word segmentation is also thought to be necessary for Chinese text categorization [16], but we avoid the need again by considering character level language models.

For Chinese topic detection we considered a data set investigated in [16]. The corpus in this case is a subset of the TREC-5 data set created for research on Chinese text 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) and the six most frequent groups were selected to make a Chinese text categorization data set, as shown in Table 11. Following [16, 17], we converted the problem into 6 binary classification problems. In each case, we randomly selected 500 positive examples and then selected 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.

Code	Topic
0	Politics, Law and Society
1	Literature and Arts
2	Education, Science and Culture
3	Sports
4	Theory and Academy
5	Economics

Table 11: Chinese topics

Table 12 shows the results of the character level language modeling classifiers. Table 13 shows the results of using an SVM classifier, where the first column is the number of selected features based on frequency ranking. (We use the SVM^{light} [23] toolkit with linear kernels.) The entries are micro-average F-measure. We observe a performance of over **86.7%** for this task, using bi-gram (2 Chinese characters) or higher order models. Standard naive Bayes model (n=1 and Laplace smoothing) achieves 85.7% performance. SVM achieves stable performance of about 81.5%. He et al. [16] reported a performance of about 82% using SVM based on word segmentation and feature selection.

	Absolute	Laplace	Good-Turing	Linear	Witten-Bell
1	0.855	*0.857*	0.858	0.859	0.859
2	**0.867**	0.798	0.862	0.860	0.859
3	0.864	0.794	0.859	0.859	0.862
4	0.866	0.801	0.860	0.863	0.862

Table 12: Results on Chinese topic detection data using character level models

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

Table 13: Results of the character level SVM classifier on Chinese data

5.4.4 Reuters-21578 data

Unlike other data sets, the Reuters-21578 data is used for multi-way classification, where each document is allowed to belong to more than one category. The data set contains 12902 Reuters news wire articles.[6] The documents are assigned to 135 topic categories. However, some categories are empty and thus there are only non-empty 118 categories, among which the 10 most frequent categories contain about 75% of the documents. There are several ways to split the documents into training and testing sets: 'ModLewis' split, 'ModApte' split, and 'ModHayes' split. The 'ModApte' train/test split is widely used in text classification research. Using the 'ModApte' split, the 10 most frequent categories and the numbers of documents used for training, testing and reserved unused in each category are listed in Table 14.

The results of micro-averaged F-measure are shown in Table 15 and Table 16 for character level models and word level models respectively.

[6]The data set is publicly available at http://www.research.att.com/˜ lewis

category	training	test	unused
earn	2877	1087	22
acq	1650	719	79
money-fx	538	179	82
grain	433	149	45
crude	389	189	54
trade	369	118	65
interest	347	131	33
wheat	212	71	23
ship	197	89	18
corn	182	56	15

Table 14: Reuters 10 most frequent categories

n	Absolute	Laplace	Good-Turing	Linear	Witten-Bell
1	0.4213	0.4207	0.4207	0.4207	0.4207
2	0.6607	0.6731	0.6790	0.6651	0.6651
3	0.7816	**0.8032**	0.7873	0.7883	0.7805
4	0.8010	0.7229	0.8024	0.7969	0.8023
5	0.7964	0.6448	0.7932	0.7855	0.7996
6	0.7832	0.6156	0.7839	0.7765	0.7884
7	0.7701	0.6131	0.7782	0.7697	0.7813

Table 15: Results on Reuters data using character level models

Word level models marginally outperform character level models. Bi-gram or tri-gram word level models are better than uni-gram models, which means that relaxing the independence assumption in naive Bayes models also helps in this case. However, in previous research, an SVM classifier obtained up to 92% accuracy [11],[7] which was based on feature selection and optimizing prior $P(c)$ for each individual category. We did not optimize the prior information and merely used a uniform prior (that is, we put equal weight for positive and negative category).

[7]They measured the performance with break even point (BEP). BEP is special case of F1 measure where recall equals precision. However, since normally it's not possible to have such a point in practice, BEP often has to be calculated using interpolation. When recall and precision values are far apart, BEP may not reflect the true behavior of the system [54, 17].

n	Absolute	Laplace	Good-Turing	Linear	Witten-Bell
1	0.7850	0.7973	0.7900	0.7880	0.7959
2	**0.8152**	0.7069	0.8124	0.8114	0.7935
3	0.8137	0.6353	0.8135	0.8105	0.7952

Table 16: Results on Reuters data using word level models

6 Analysis and discussion

The perplexity of a test document under a language model depends on several factors. The three most influential factors are the order, n, of the n-gram model, the smoothing technique used, and the number of training documents. These factors significantly influence the quality of a language model and thus may influence classification performance. We will first discuss the relationship between language modeling quality and classification performance. Then we will further analyze the influence of each individual factor. We will also discuss other factors such as the influence of prior information $P(c)$ in Equ. (20), whether character or word level models should be applied, and analyze the success and failure of the model.

6.1 Relationship between accuracy and perplexity

Figures 2 shows the relationship between classification performance and language modeling quality on the Greek authorship attribution task. (The other data sets have similar curves.) The upper part of the figure shows classification performance and the lower part shows language modeling quality measured by average entropy across all testing documents (bits per character). We can see that classification performance is almost monotonically related to language modeling quality. However, this is not absolutely true. Since our goal is to make a final decision based on the *ranking* of perplexities, not just their absolute values, a slightly superior language model in the sense of perplexity reduction (i.e. from the perspective of classical language modeling) does not necessarily lead to a better decision from the perspective of categorization accuracy.

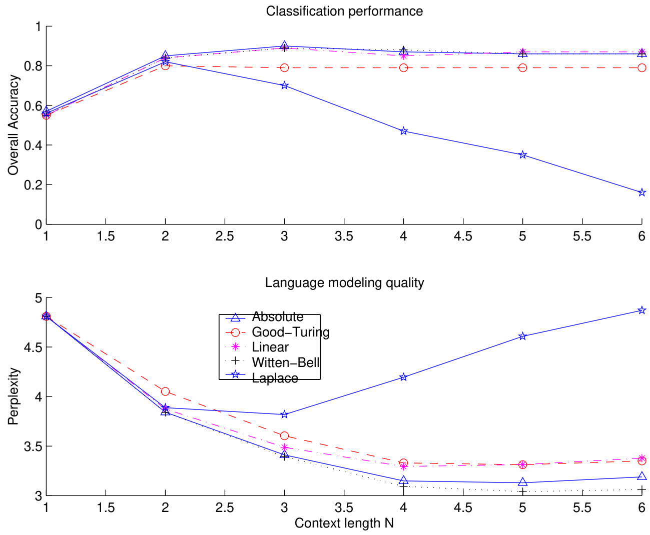

Figure 2: Relationship between classification performance and language modeling quality

6.2 Effects of n-gram order

Relaxing the naive Bayes independence assumption to consider local context dependencies is one of the main motivations of the CAN Bayes model. The order n is a key factor in determining the quality of n-gram language models. If n is too small then the model will not capture enough context. However, if n is too large then this will create severe sparse data problems. Both extremes result in a larger perplexity than the optimal context length and decrease classification performance. Figures 3 illustrates the influence of order n on classification performance in the previous five experiments (Greek authorship attribution, Greek text genre classification, English topic detection, Japanese topic detection and Chinese topic detection) using absolute smoothing. From the curves, one can see that as the order increases, classification accuracy increases, presumably because the longer context better captures the regularities of the text. However, at some point accuracy begins to decrease and entropy begins to increase as sparse data problems begin to set in. Interestingly, the effect is more pronounced in some experiments (Greek genre classification) but less so in other experiments (topic detection under any language). The sensitivity demonstrated in the Greek genre case could still be attributed to the sparse data problem (over-fitting in genre classification could be more serious than other problems). Data sparseness also explains why the context information does not help in the Chinese data beyond 2-grams. The performance increases 3 to 4% from 1-gram to

2-gram models, but does not increase any more. There are 3573 most commonly used characters in Chinese compared to 100 (upper case, lower case, and punctuation) in English. Data sparseness is much more serious in Chinese than in English. Also, most Chinese words consist of 1 to 2 characters. Bi-grams have been found very effective in Chinese IR [29]. However, longer context information does not help significantly improve classification accuracies in Greek and English at the word level (except for Witten-Bell smoothing, which may be due to its poor performance on uni-gram models). Bi-gram models generally outperform uni-gram models. However, the over-fitting problem in word level models set in earlier because the sparse data problem at the word level is much more serious than at the character level.

Overall, relaxing independence assumptions can help in cases where there is sufficient training data to reduce the sparse data problem, such as with character level models. However, when one does not have sufficient training data, it becomes important to use short context models that avoid sparse data problems. These results may offer another explanation of why the naive Bayes classifier is preferred in practice, because normally one can not get sufficient labeled training data to train a large-context model.

The optimal order n of n-gram models depends on language and task, which in practice can be determined by cross validation.

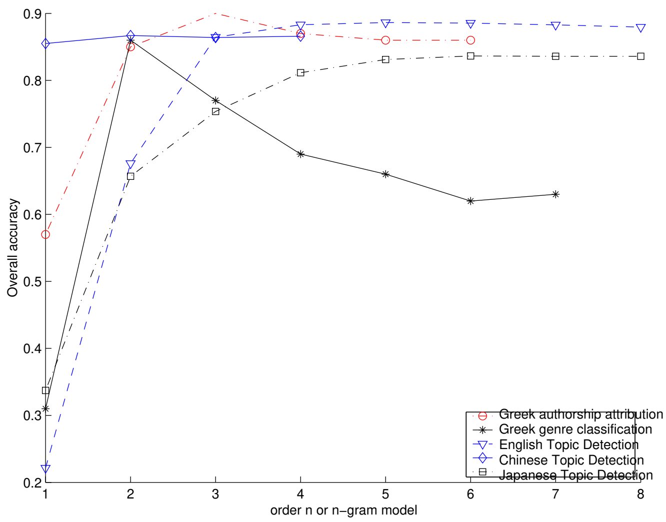

Figure 3: Influence of the order n on classification performance

6.3 Effects of smoothing technique

Another key factor affecting the performance of a language model is the smoothing technique used. We illustrate the results on the 20 Newsgroup data sets with character level and word level models in Figure 4 and Figure 5 respectively (other data sets have similar curves).

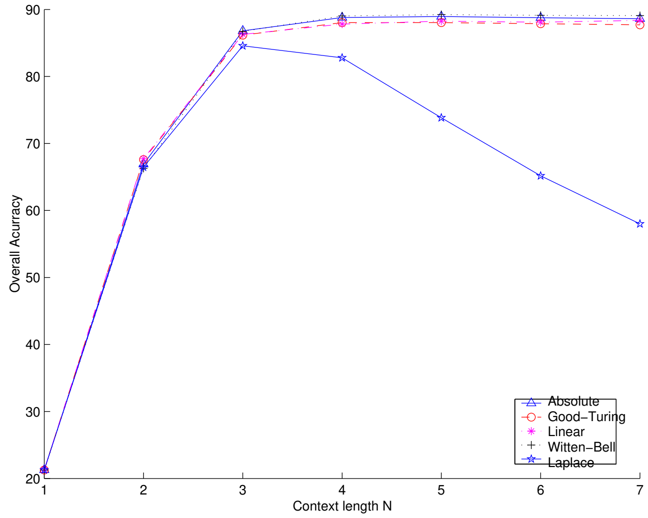

Figure 4: Results on 20 newsgroup data using character level models

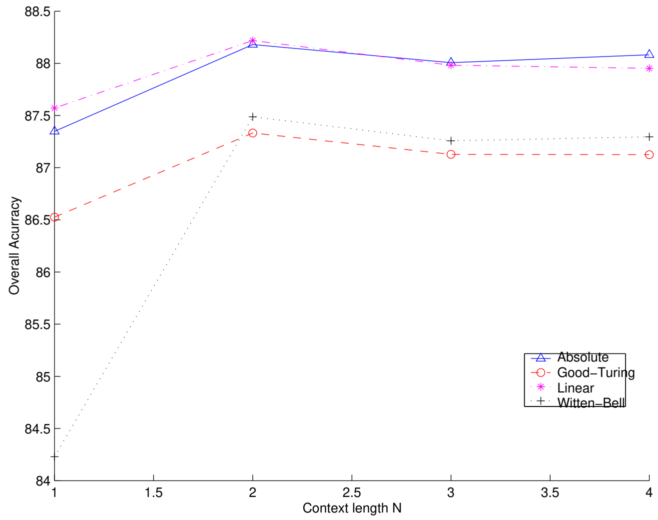

Figure 5: Results on 20 newsgroup data using word level models (Laplace smoothing omitted for clarity)

Here we find that Laplace smoothing over-fits very quickly. Overall, Laplace smoothing is not a good choice for smoothing naive Bayes models. Other smoothing techniques perform similarly on character models but vary significantly on word level models. On word level models, absolute smoothing and linear smoothing outperform other smoothing techniques. However, in character based models, for the most part, one can use any standard smoothing technique (except Laplace smoothing) and obtain comparable performance. One reason that the smoothing technique does not make a big difference for character based models is that character based models have a very small vocabulary.

6.4 Influence of training size

Clearly, the size of the training corpus can affect the quality of a language model. Normally with a larger training corpus more reliable statistics can be obtained which leads to better prediction accuracy on test data. To test the effect of training set size we obtained an additional 10 documents from each author in the Greek data set. In fact, this same additional data has been used in [47] to improve the accuracy of their method from 72% to 87%. Here we find that the extra training data also improves the accuracy of our method, although not so dramatically.

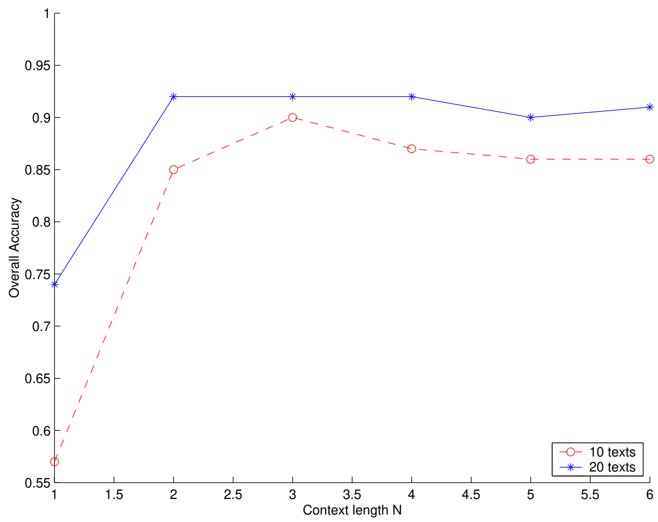

Figure 6: Influence of training size on accuracy

Figure 6 shows the improvement obtained for n-gram language models using absolute smoothing. Here we can see that indeed extra training data uniformly improves attribution accuracy. On the augmented training data the best model (tri-gram) now obtains a **92%** attribution accuracy, compared to the 90% we obtained originally. Moreover, this improves the best result obtained in [47] of 87%. However, our improvement (90% to 92%) is not nearly as great as that obtained by [47]

(71% to 87%). However, this could be due to the fact that it is harder to reduce a small prediction error. A similar phenomenon also occurs with other smoothing techniques.

6.5 Choice of priors

The prior $P(c)$ in Equ.(20) can be empirically computed from training data, or sometimes can be assumed to be uniform if we do not know its true distribution. In the Japanese topic detection experiments, the data had 24 categories and documents were distributed unevenly between categories (with a minimum of 1747 and maximum of 53975 documents per category). This imbalanced distribution could cause some difficulty if one assumed a uniform prior over categories. However, with an empirical prior distribution learned from training data, we observe that the results do not change significantly from simply using a uniform prior. The comparison of using a uniform prior versus an empirical prior is shown in Table 17 (with absolute smoothing). There is a significant improvement for uni-gram models when using an empirical instead of uniform prior. However, the difference becomes smaller and eventually can be ignored as higher order models are used. Basically, the direct empirical distribution computed from training data is not very helpful in text classification tasks. However, optimizing the priors based on held out data sets are helpful [11, 49].

n	Uniform prior	Empirical prior
1	0.33	0.38
2	0.66	0.67
3	0.75	0.76
4	0.81	0.81
5	0.83	0.83
6	0.84	0.84
7	0.84	0.84
8	0.83	0.83

Table 17: Uniform prior versus empirical prior on Japanese topic detection results

6.6 Character level versus word level

For many Asian languages such as Chinese and Japanese, where word segmentation is hard, character level CAN Bayes models are well suited for text classification because they avoid the need for word segmentation. For Western languages such as Greek and English, one can work at both the word and character levels. Table 18 compares character and word level results on the Greek and English experiments.

Task	Character level	Word level
Greek authorship attribution	90%	96%
Greek genre detection	86%	81%
English topic detection (20news)	89.1%	88%

Table 18: Character level versus word level results

In the Greek authorship attribution task and movie review sentimental classification task, word level models significantly outperform character level models. However, in other experiments, character level models outperform word level models. It seems that for informal texts, such as Newsgroup data and other on-line data, character models have an advantage since that they can capture some regularities that word level models miss in this case, such as spelling errors. By relaxing the context, character level models can also capture regularities at the word level, and even phrase level regularities. However, it is not clear whether a character or word model should be used for a specific text. Perhaps combining the two levels could result in more robust and more accurate results.

6.7 Overall assessment

Most of the results we have reported here are comparable to (or much better than) the start of the art results on the same data sets. For example, our 90% accuracy for Greek authorship attribution is much better than the 72% reported in [46]. Our 86% Greek genre classification accuracy is better than the 82% reported in [47] which is based on a much more complicated analysis. Our 89.13% accuracy on the 20 Newsgroups data set is better than the best result, 87.5%, reported in [40] which is based on a combination of SVMs and error correcting output coding (ECOC). Our 86.7% accuracy on the Chinese TREC data is better than the 81.7% achieved by SVMs. Overall, the chain augmented naive Bayes classifier works very well, even though it is a much simpler technique than these other methods and it is not specialized to any particular data set.

The success of this simple method, we think, is due to the effectiveness of well known statistical language modeling techniques, which surprisingly have had little significant impact on the learning algorithms normally applied to text categorization. Nevertheless, statistical language modeling is also concerned with modeling the semantic, syntactic, lexicographical and phonological regularities of natural language—and would seem to provide a natural foundation for text categorization problems. One interesting difference, however, is that instead of explicitly pre-computing features and selecting a subset based on arbitrary decisions, the language modeling approach simply considers all character (or word) subsequences occurring in the text as candidate features, and implicitly considers the contribution of *every* feature in the final model. Thus, the language modeling approach completely avoids a potentially error-prone feature selection process.

However, language modeling based approach does not win in all cases. We also found that in the Reuters-21578 data set, SVMs still significantly outperformed our language modeling based classifiers. In this case, the reason appears to be that the sparse data problem in this data set is very severe in some categories (such as corn, ship, and wheat). Here, a traditional SVM classifier with an explicit feature selection procedure may have an advantage since the feature selection procedure may be able to select a few words which are sufficient to distinguish the categories, whereas the language modeling approach does not have enough data to obtain reliable probability estimates. This suggests that straightforwardly applying language model based classifiers to multi-way classification might be problematic. However, in [49] a compression based language modeling approach was applied to the same data set and comparable performance was achieved by optimizing the parameters. Nevertheless, language modeling based classifiers still improve plain naive Bayes classifiers in this case.

We found that the performance of a classifier depends both on the language and the task. For example, SVMs achieve state of the art performance on Reuters data. However, they do not perform well on Chinese and Japanese classification tasks (see our experiments, also see He et al. [16]). SVMs do not perform significantly better than naive Bayes on the 20 Newsgroup data either [40] Overall, our findings suggest that different classifiers are superior for different languages and tasks, although character level n-gram models are general and robust.

7 Relation to previous research

7.1 Relaxing the naive Bayes independence assumption

Relaxing the independence assumption of the Naive Bayes model is a much researched idea in machine learning, and other researchers have considered learning tree augmented naive Bayes classifiers from data [14, 25]. However, learning the hidden tree structure is problematic, and our chain augmented naive Bayes model, being a special case of TAN Bayes, better preserves the simplicity of the naive Bayes classifier while introducing some of the additional dependencies of TAN Bayes. The extra complexity introduced by the CAN model is mainly on extracting the n-grams and their frequencies from the training corpus, which can be done efficiently in a single pass. Work on considering context information for learning text classifiers also include context sensitive rule learning and on-line learning [7].

7.2 N-gram models in information retrieval

N-gram models have been extensively used in text retrieval, particularly in Asian language text retrieval, such as Chinese and Japanese [28]. N-gram models are also used in English information retrieval for phrase weighting in IR [51]. In text classification, n-gram models have also been previously investigated [4, 8, 21]. However, they have typically used n-grams as features for a traditional feature selection process, and then deployed classifiers based on calculating feature vector similarities. In our CAN Bayes models, we do not perform explicit feature selection at all (although of course we have to choose a few factors, such as the order n, smoothing technique and vocabulary size). Thus, in our case, all n-grams remain in the model, and their importance is implicitly considered by their contribution to perplexity.

7.3 Language models for text classification

In principle, any language model can be used to perform text classification based on Equ. (18). However, n-gram models are extremely simple and have been found to be effective in many applications. Interestingly, language modeling research has been conducted in parallel and mostly independently in the speech recognition and text compression communities. In text compression, character level n-gram models are widely used, e.g., the PPM (predication by partial matching) model [2]. The PPM model has been a benchmark in text compression for decades. However, in speech recognition, research has focused on word level n-gram models.

These different tasks have motivated different strategies for smoothing. For example, the PPM model deals with out of vocabulary words with an escape method, which works in a fashion similar to the back-off model although essentially a weighted linear interpolation n-gram models [2, 48].[8] However, in statistical language modeling research, many other different smoothing strategies have been proposed, as briefly discussed in Section 3. It has been found that "Witten-Bell smoothing" (or PPMC) does not perform well on word level language modeling (although we also find that it performs well at character level) and other smoothing techniques such as Good-Turing and absolute smoothing perform better. (An interesting question is how these smoothing techniques will perform in text compression, although this is irrelevant to this paper.)

PPM model have recently been found to be effective in text mining [53] and promising results were also obtained in text classification [49]. A PPM model is trained incrementally as text is read in. Building such an adaptive PPM model is expensive however [2], and our back-off models are relatively much simpler. Using compression techniques for text classification has also been investigated in [3], where the authors seek a model that yields the minimum compression rate increase when a new test document is introduced. However, this method is found not to be generally effective nor efficient [15]. In our approach, we evaluate the perplexity (or entropy) directly on test documents, and find the outcome to be both effective and efficient.

8 Conclusion and Future work

We have presented a chain augmented naive Bayes classifier (CAN) based on statistical n-gram language modeling. Our CAN Bayes model captures dependence between adjacent attributes as a

[8] The smoothing method "C" used in [52], namely PPMC, has often been mistakenly referred to as "Witten-Bell smoothing", although it was originally invented by Alistair Moffat [36]. It has been found that PPMC and its variants PPMD [20], PPM* [48], perform well in text compression.

Markov chain. By using better smoothing techniques than Laplace smoothing we obtain further performance improvements. Our CAN Bayes modeling approach avoids explicit feature selection by considering *every* feature in the model and measure their importance implicitly. The CAN model is able to work at either the character level or the word level, which provides language independent abilities to handle Eastern languages like Chinese and Japanese just as easily as Western languages like English or Greek.

The approach is evaluated on four different languages and three different text classification problems. Surprisingly, we obtain state of the art or better performance in most cases. We have experimentally analyzed the influence of different factors that can affect the accuracy of this approach, and found that for the most part the results are robust to perturbations of the basic method. We have also analyzed the successes and failures of the model. To us, these results suggest that basic statistical language modeling ideas might be more relevant to other areas of natural language processing than commonly perceived.

We are currently extending the CAN model in many aspects. In particular, the current model is trained in a supervised fashion, which requires enough labeled training documents to obtain good performance. However, labeled training data is not easy to obtain. An important issue is how to extend it to incorporate unlabeled data. Since the character level CAN model avoids word segmentation problems in Asian languages such as Chinese and Japanese, it is worthwhile to further investigate its performance on these languages compared to other machine learning techniques.

9 Acknowledgments

We thank E. Stamatatos for supplying us with the Greek data sets, Ji He for the Chinese data set, and Akiko Aizawa for the Japanese data. We are grateful to the three anonymous reviewers for their detailed comments and the four anonymous reviewers for their constructive comments on the earlier conference version of this paper. This work was also supported in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

[1] A. Aizawa. Linguistic Techniques to Improve the Performance of Automatic Text Categorization. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposim (NLPRS)*, pages 307–314, 2001.

[2] T. Bell, J. Cleary, and I. Witten. *Text Compression*. Prentice Hall, 1990.

[3] D. Benedetto, E. Caglioti, and V. Loreto. Language Trees and Zipping. *Physical Review Letters*, 88, 2002.

[4] W. B. Cavnar and J. M. Trenkle. N-Gram-Based Text Categorization. In *3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR)*, 1994.

[5] S. Chen and J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.

[6] K. Church and W. Gale. A Comparison of the Enhanced Good-Turing and Deleted Estimation Methods for Estimating Probabilities of English Bigrams. *Computer Speech and Language*, 5:1, 1991.

[7] W. Cohen and Y. Singer. Context-sensitive Learning Methods for Text Categorization. *ACM Transactions on Information Systems*, 17:141–173, 1999.

[8] M. Damashek. Gauging Similarity with N-Grams: Language-Independent Categorization of Text? *Science*, 267:843 – 848, 1995.

[9] P. Domingos and M. Pazzani. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. *Machine Learning*, 29:103–130, 1997.

[10] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[11] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Feature Engineering for Text Classification. In *Proceedings of ACM Seventh International Conference on Information and Knowledge Management (CIKM)*, pages 148–155, 1998.

[12] S. Eyheramendy, D. Lewis, and D. Madigan. On the Naive Bayes Model for Text Categorization. In *Proceedings Artificial Intelligence & Statistics 2003*, 2003.

[13] M. Federico and R. De Mori. Language modelling. In Renato De Mori, editor, *Spoken Dialogues with Computers*, chapter 7. Academy Press, London, UK, 1998.

[14] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29:31–163, 1997.

[15] J. Goodman. Comment on Language Trees and Zipping. Unpublished Manuscript, 2002.

[16] J. He, A. Tan, and C. Tan. A Comparative Study on Chinese Text Categorization Methods. In *Proceedings of PRICAI'2000 International Workshop on Text and Web Mining*, pages 24–35, 2000.

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

[18] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, Centre for Telematics and Information Technology, University of Twente, 2001.

[19] D. Holmes and R. Forsyth. The Federalist Revisited: New Directions in Authorship Attribution. *Literary and Linguistic Computing*, 10:111–127, 1995.

[20] P.G. Howard. *The Design and Analysis of Efficient Loseless Data Compression Systems*. PhD thesis, Brown University, 1993.

[21] Stephen Huffman. Acquaintance: Language-Independent Document Categorization by N-grams. In *In Donna K. Harman and Ellen M. Voorhees, editors, Proceedings of TREC-4, 4th Text Retrieval Conference*, pages 359 – 371, 1995.

[22] F. Jelinek. Self-Organized Language Modeling for Speech Recognition. In Alex Weibel and Kay-Fu Lee, editors, *Readings in Speech Recognition*, pages 450–505. Morgan Kaufmann, Los Altos, CA, 1990.

[23] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, 1998.

[24] S. Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(3):400–401, March 1987.

[25] E. Keogh and M. Pazzanni. A Comparison of Distribution-based and Classification-based Approaches. In *Proceedings Artificial Intelligence & Statistics 1999*, 1999.

[26] B. Kessler, G. Nunberg, and H. Schüze. Automatic Detection of Text Genre. In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics (ACL)*, 1997.

[27] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, S peech and Signal Processing*, volume 1, pages 181–184, Detroit, MI, 1995.

[28] K. L. Kwok. Comparing Representations in Chinese Information Retrieval. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 34–41. ACM, 1997.

[29] K. L. Kwok. Employing Multiple Representations for Chinese Information Retrieval. *Journal of the American Society for Information Science (JASIS)*, 50(8):709–723, 1999.

[30] K. Lang. Newsweeder: Learning to Filter Netnews. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML)*, pages 331–339, 1995.

[31] Y. Lee and S. Myaeng. Text Genre Classification with Genre-Revealing and Subject-Revealing Features. In *Proceedings of The 25th Annual International ACM SIGIR Conference on Research and Development in Information (SIGIR)*, 2002.

[32] D. Lewis. *Representation and Learning in Information Retrieval.* PhD thesis, Computer Science Deptment, University of Massachusetts, 1992.

[33] D. Lewis. Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In *Proceedings 10th European Conference on Machine Learning (ECML)*, 1998.

[34] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, Massachusetts, 1999.

[35] A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *Proceedings of AAAI-98 Workshop on "Learning for Text Categorization*, 1998.

[36] A. Moffat. Implementing the PPM Data Compression Scheme. *IEEE Transactions on Communications*, 31(11):1917–1921, 1990.

[37] H. Ney, U. Essen, and Kneser R. On Structuring Probabilistic Dependencies in Stochastic Language Modeling. *Computer Speech and Language*, 8(1):1–28, 1994.

[38] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, 2002.

[39] J. Ponte and W. Croft. A Language Modeling Approach to Information Retrieval. In *Proceedings of ACM Research and Development in Information Retrieval (SIGIR)*, pages 275–281, 1998.

[40] J. Rennie. Improving Multi-class Text Classification with Naive Bayes. Master's thesis, M.I.T., 2001.

[41] I. Rish. An Empirical Study of the Naive Bayes Classifier. In *Proceedings of IJCAI-01 Workshop on Empirical Methods in Artificial Intelligence.*, 2001.

[42] S. Robertson and K. Sparck Jones. Relevance Weighting of Search Terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[43] J. C. Schmitt. Trigram-based Method of Language Identification. October 1991. U.S. Patent No. 5,062,143.

[44] S. Scott and S. Matwin. Feature Engineering for Text Classification. In *Proceedings of The Sixteenth International Conference on Machine Learning (ICML)*, pages 379–388., 1999.

[45] F. Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[46] E. Stamatatos, N. Fakotakis, and G. Kokkinakis. Automatic Text Categorization in Terms of Genre and Author. *Computational Linguistics*, 26(4):471–495, 2000.

[47] E. Stamatatos, N. Fakotakis, and G. Kokkinakis. Computer-based Authorship Attribution without Lexical Measures. *Computers and the Humanities*, 35:193–214, 2001.

[48] W. Teahan. *Modelling English Text*. PhD thesis, University of Waikato, 1998.

[49] W. Teahan and D. Harper. Using Compression-Based Language Models for Text Categorization. In *Proceedings of Workshop on Language Modeling and Information Retrieval (LMIR)*, 2001. (also appear in Language Modeling and Information Retrieval, Kluwer, 2003).

[50] P. Turney. Thumbs Up or Thumbs Down? Semantic Oritentation Applied to Unsupervised Classification of Reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

[51] A. Turpin and A. Moffat. Statistical Phrases for Vector-Space Information Retrieval. In *22th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 1999.

[52] I. Witten and T. Bell. The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression. *IEEE Transactions on Information Theory 37(4)*, 37(4), 1991.

[53] I. Witten, Z. Bray, M. Mahoui, and T. Teahan. Text mining: A New Frontier for Lossless Compression. In *Proceedings of IEEE Data Compression Conference 1999*, 1999.

[54] Y. Yang. An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, 1(1/2):67–88, 1999.

[55] H. Zhang and C. Ling. Learnability of Augmented Naive Bayes in Nominal Domains. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, 2001.