Combining Naive Bayes and n-Gram Language Models for Text Classification

Fuchun Peng Dale Schuurmans

School of Computer Science, University of Waterloo
200 University Avenue West, Waterloo, Ontario, Canada, N2L 3G1
{f3peng,dale}@cs.uwaterloo.ca

Abstract. We augment the naive Bayes model with an n-gram language model to address two shortcomings of naive Bayes text classifiers. The *chain augmented* naive Bayes classifiers we propose have two advantages over standard naive Bayes classifiers. First, a chain augmented naive Bayes model relaxes 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, smoothing techniques from statistical language modeling can be used to recover better estimates than the Laplace smoothing techniques usually used in naive Bayes classification. Our experimental results on three real world data sets show that we achieve substantial improvements over standard naive Bayes classification, while also achieving state of the art performance that competes with the best known methods in these cases.

1 Introduction

Naive Bayes classifiers have been proven successful in many domains, despite the simplicity of the model and the restrictiveness of the independence assumptions it makes. Domingos and Pazzanni [5] point out 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 [14], and it has been shown in practice that functionally dependent attributes can indeed improve classification accuracy in some cases [8, 23].

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) [8] 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 [11, 29], 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. In this paper, we augment the naive Bayes text classifier by including attribute 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.

In addition to proposing the CAN model, we also investigate the 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 smoothing techniques from statistical language modeling in text classification applications. For example, Laplace smoothing does not consider the possibility of encountering new words in testing that were not observed during training, yet one can often do better by explicitly considering the possibility of such out of vocabulary (OOV) terms. We consider several smoothing techniques that have been developed for handling OOV values.

Below we first describe the naive Bayes classifier (Section 2) and then present the basic n-gram language model (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 three real world data sets in Section 5, and conclude with a brief discussion in Section 6.

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[1] is based on a simple application of *Bayes' rule* [6, chapter 10]:

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

To simplify the presentation, we re-write Eq. (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 [25].

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[2] $d = (v_1, v_2, ...v_K)$. 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 Eq. (2) to

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

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

$$\begin{aligned} c^* &= \arg\max_{c \in C} \; \{P(c|d)\} && (4) \\ &= \arg\max_{c \in C} \left\{ P(c) \times \frac{\prod_{j=1}^{K} P(v_j|c)}{P(d)} \right\} && (5) \\ &= \arg\max_{c \in C} \left\{ P(c) \times \prod_{j=1}^{K} P(v_j|c) \right\} && (6) \end{aligned}$$

Note that going from Eq. (5) to Eq. (6) is valid because $P(d)$ is a constant for every category c. A MAP classifier (Eq. (4)) is optimal in the sense of minimizing *zero-one* loss (misclassification error). If the independence assumption holds, then a classifier based on Eq. (6) is also optimal [6].

The prior distribution $P(c)$ can also be used to incorporate additional assumptions. Two commonly used prior distributions are Dirichlet distribution and uniform distribution. When uniform distribution is used as the prior, the MAP classifier becomes equivalent to the maximum likelihood (ML) classifier.

$$c^* = \arg\max_{c \in C} \left\{ \prod_{j=1}^{K} P(v_j|c) \right\} \tag{7}$$

Eq. (7) is called the *maximum likelihood* naive Bayes classifier. 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 [7]. It has been shown that for text categorization applications, the multinomial model is most often the best choice [7, 16], therefore we will only consider

[2] Attributes are also called features in many papers. Feature selection is an important procedure in many classifiers [17, 24].

the multinomial naive Bayes model in this paper. Fig. 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.[3]

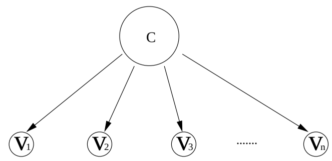

Fig. 1. Graphical model of a naive Bayes classifier

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

$$P(D^c|\Theta) = \frac{N^c!}{\prod_j N_j^c!} \prod_j (\theta_j^c)^{N_j^c} \tag{8}$$

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

$$\theta_j^c = \frac{N_j^c}{N^c} \tag{9}$$

Note that Eq. (9) 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:

$$\theta_j^c = \frac{N_j^c + a_j}{N^c + a} \tag{10}$$

where $a = \sum_j a_j$. A special case of Laplace smoothing is *add one* smoothing [15, chapter 6] obtained by setting $a_j = 1$.

Unfortunately, although Eq. (10) considers some attribute values that do not occur in D^c, it does not consider any attribute value that never occurs in any training document (i.e. OOV values), since $\sum_j \theta_j^c = 1$. That is, Laplace smoothing simply ignores all attribute values that fall out of the selected vocabulary. This ignores some information that could be provided by the OOV values which might help distinguish documents. However, we can create a pseudo attribute to represent all OOV attributes and then use Laplace smoothing to deal with OOV words. This is one of the techniques we use in our experiments below. Several

[3] Note that the figure does not include frequency information, as in [7], because here we only consider the multinomial naive Bayes model, and thereby avoid confusion with the binomial naive Bayes model.

other smoothing techniques have been developed in statistical language modeling that consider the effect of OOV words by reserving some probability mass for such words (i.e., $\sum_j \theta_j^c < 1$). Below, we show that 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 Markov 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 [10, 13, 21].

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 on this corpus

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

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

$$P(w_1 w_2 ... w_T) = \prod_{i=1}^{T} P(w_i|w_1...w_{i-1}) \tag{11}$$

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

$$P(w_i|w_{i-n+1}...w_{i-1}) = \frac{\#(w_{i-n+1}...w_i)}{\#(w_{i-n+1}...w_{i-1})} \tag{12}$$

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

$$P(w_i|w_{i-n+1}...w_{i-1}) \quad = \begin{cases} \hat{P}(w_i|w_{i-n+1}...w_{i-1}), \\ \qquad \text{if } \#(w_{i-n+1}...w_i) > 0 \\ \beta(w_{i-n+1}...w_{i-1}) \times P(w_i|w_{i-n+2}...w_{i-1}), \\ \qquad \text{otherwise} \end{cases} \tag{13}$$

where

$$\hat{P}(w_i|w_{i-n+1}...w_{i-1}) = \frac{discount\,\#(w_{i-n+1}...w_i)}{\#(w_{i-n+1}...w_{i-1})} \tag{14}$$

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

$$\beta(w_{i-n+1}...w_{i-1}) = \frac{1 - \sum_{x\in(w_{i-n+1}...w_{i-1}x)} \hat{P}(x|w_{i-n+1}...w_{i-1})}{1 - \sum_{x\in(w_{i-n+1}...w_{i-1}x)} \hat{P}(x|w_{i-n+2}...w_{i-1})} \tag{15}$$

The discounted probability (14) can be computed using different smoothing approaches including linear smoothing, absolute smoothing, Good-Turing smoothing and Witten-Bell smoothing [2]. Note that Laplace smoothing can not be applied in n-gram language modeling with $n > 1$ because it does not allow for unseen n-grams, but these are almost certain to occur in test documents.

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.

To allow a comparison to Laplace smoothing (Eq. (10)), we briefly describe the standard smoothing techniques used in n-gram language modeling.

Absolute smoothing Here, the frequency of a word is subtracted by a constant b, so the discounted probability (Eq. 14) 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 the upper bound $b = \frac{n_1}{n_1+2n_2}$, and n_i is the number of events which occur *exactly* i times in training data [18]. This notation also applies to other smoothing techniques.

Linear smoothing Here, the discounted probability is calculated as

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

where Z is the number of uni-grams, which corresponds to the number of words in the training data [18].

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

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

Witten-Bell smoothing Witten-Bell smoothing is very similar to Laplace smoothing, except it reserves probability mass for OOV values, whereas Laplace smoothing does not. Here the discounted probability is calculated as

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

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

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

4 Using 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. In this case

$$\begin{aligned} c^* = \arg\max_{c \in C} \{P(c|d)\} \quad &= \quad \arg\max_{c \in C} \{P(d|c)P(c)\} && (16) \\ = \arg\max_{c \in C} \{P(d|c)\} & && (17) \\ = \arg\max_{c \in C} \left\{ \prod_{i=1}^{T} P(w_i|w_{i-n+1}...w_{i-1}, c) \right\} & && (18) \end{aligned}$$

where the step from Eq. (16) to Eq. (17) assumes a uniform prior over categories, and the step from Eq. 17 to Eq. (18) uses the Markov n-gram independence assumption. The principle for using an n-gram language model as a text classifier is to determine the category that makes a given document *most likely* to have been generated by the category model (Eq. (18)). Thus, we train a separate language model for each category, and classify a new document by evaluating its likelihood under each category, choosing the category according to Eq. (18).

A graphical model of a bi-gram augmented naive Bayes text classifier is given in Fig. 2. A comparison of Fig. 1 and Fig. 2 shows that the standard naive Bayes classifier and n-gram augmented naive Bayes classifier differ, in that the latter introduces an additional Markov chain dependence between adjacent attributes. Due to this similarity, we refer the n-gram augmented naive Bayes classifier as a **C***hain* **A***ugmented* **N***aive Bayes* classifier (CAN).

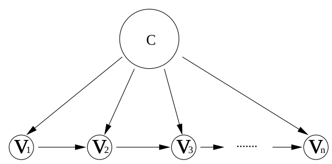

Fig. 2. Graphical model of a bi-gram chain augmented naive Bayes classifier

We now experimentally evaluate the performance of chain augmented naive Bayes classifier on three real word data sets.

5 Empirical Evaluation

We present experimental results on different text classification problems in three different languages. First, we consider a Greek authorship attribution problem, where Greek documents must be classified as being written by one of 10 modern Greek authors. Then we consider an English newsgroup topic detection problem, where documents must be classified as belonging to one of 20 newsgroups. Finally, we consider a Chinese TREC topic detection problem, where Chinese documents must be classified into one of 6 topics.

5.1 Data Sets and Performance Measure

The Greek authorship attribution data we used has been previously investigated in [26], and can be downloaded from the WWW site of the Modern Greek weekly newspaper "TO BHMA". The corpus consists of two document collections containing 20 works from 10 different authors (totaling 200 documents) each. In our experiments, we divided the collections into 10 training and 10 testing documents from each author. (The authors used in the data sets are shown in Table 1.) We also considered any concatenated character string delimited by white-space to be a word, which introduces some noise since many character strings are concatenated with punctuation marks. Nevertheless, we did not perform any extra pre-processing to remove this noise. The number of unique words observed in the training corpus is 31,502. From these, we select the most frequent 30,000 words to form the vocabulary, which comprises 98.7% of the total word occurrences.

The English 20 Newsgroup data[4] we used has been used investigated in research on text categorization [7, 16, 19, 22]. It contains 19,974 non-empty documents evenly distributed across 20 newsgroups, which form our categories. We randomly select 80% of the documents to be used for training and leave the

[4] Available at http://www.ai.mit.edu/˜ jrennie/20Newsgroups/

code	Author Name	code	Author Name
0	S. Alaxiotis	5	D. Maronitis
1	G. Babiniotis	6	M. Ploritis
2	G. Dertilis	7	T. Tasios
3	C. Kiosse	8	K. Tsoukalas
4	A. Liakos	9	G. Vokos

Table 1. Authors in the Greek data set

remaining 20% for testing. In our experiments, we selected the 70,000 most frequent words to form the vocabulary, which comprises 92.3% of total word occurrences. As with the Greek data set, the selected words have some noise which we did not remove.

code	Topic	code	Topic
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 2. Topics in the 20 Newsgroup data set

Finally, the Chinese data set we used has been previously investigated in [9]. 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 77,733 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, He et al. [9] first clustered documents 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, as shown in Table 3. In each group, 500 documents were randomly selected for training and 100 documents were reserved for testing. In our experiments, we selected the most frequent 2,000 Chinese characters to serve as the vocabulary, which comprised over 99% of the total character occurrences in the training corpus.

In each of our experiments, we measured categorization performance by the *overall accuracy*, which is the number of correctly identified texts divided by the total number of texts considered [28].

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 3. Topics in the Chinese TREC data set

5.2 Experimental Results

The results of Greek authorship attribution are shown in Table 4. The upper half shows the results when considering out of vocabulary (OOV) words, and the lower half shows the results without considering OOV words. The OOV rate is the average rate of words that are out of vocabulary on all testing documents. The first column is the order of n-gram model, and each other column represents a different smoothing technique. Since Laplace smoothing can not be applied to n-gram models with $n > 1$, the results on those entries are marked as "N/A".

Considering OOV words					
n	Adding One	Absolute	Good-Turing	Linear	Witten-Bell
1	0.7600	0.9600	0.9300	0.9600	0.6100
2	N/A	0.9600	0.9500	0.9500	0.8300
3	N/A	0.9600	0.9500	0.9500	0.8300
4	N/A	0.9600	0.9500	0.9500	0.8400
Not considering OOV words : 0.2277 OOV rate					
1	0.6900	0.9400	0.9300	0.9100	0.5500
2	N/A	0.9300	0.9300	0.8900	0.8200
3	N/A	0.9500	0.9300	0.8900	0.8300
4	N/A	0.9500	0.9300	0.8900	0.8300

Table 4. Results on Greek authorship attribution

The results of topic detection on the English 20 Newsgroup data set are shown in Table 5.

Chinese topic detection is more difficult because words are not white-space delimited, as they are in English and Greek. Normally, a word segmenter is required for Chinese text classification [9]. However, we avoid the need for explicit segmentation by using a *character level* CAN Bayes classifier. Table 6 shows our experimental results on Chinese topic detection.

For Greek and English, we can also apply character level CAN Bayes classifiers. The results of character level models on Greek and English are shown in Table 7 and Table 8 respectively. The size of the character vocabulary is 150 for Greek and 100 for English.

Considering OOV words					
n	Adding One	Absolute	Good-Turing	Linear	Witten-Bell
1	0.8505	0.8664	0.8595	0.8720	0.8449
2	N/A	0.8796	0.8778	0.8796	0.8749
3	N/A	0.8751	0.8741	0.8733	0.8693
4	N/A	0.8757	0.8727	0.8743	0.8704
Not considering OOV words : 0.1121 OOV rate					
1	0.8431	0.8717	0.8611	0.8727	0.8375
2	N/A	0.8804	0.8791	0.8794	0.8733
3	N/A	0.8757	0.8746	0.8765	0.8704
4	N/A	0.8780	0.8757	0.8770	0.8704

Table 5. Topic detection results on the English 20 Newsgroup data with word models

Considering OOV words					
n	Adding One	Absolute	Good-Turing	Linear	Witten-Bell
1	0.7650	0.7667	0.7600	0.7567	0.7667
2	N/A	0.8050	0.8050	0.7950	0.8017
3	N/A	0.8017	0.8133	0.8000	0.8000
4	N/A	0.8017	0.8067	0.8067	0.8000
Not considering OOV words : 0.0514 OOV rate					
1	0.7750	0.7783	0.7783	0.7783	0.7783
2	N/A	0.7967	0.7933	0.7850	0.7933
3	N/A	0.7883	0.7933	0.7883	0.7883
4	N/A	0.7783	0.7883	0.7883	0.7833

Table 6. Topic detection results on Chinese TREC data with character models

5.3 Discussion

We now make some observations based on the above results.

Effect of considering OOV words As mentioned in the introduction, Laplace smoothing does not consider OOV values, and hence loses any information provided by these values. This weakness is revealed in our experiments. By comparing the upper parts with the lower parts of the tables, one can see that when the OOV rate is low (for example, Table 5 and 6), the performance obtained when ignoring OOV words is roughly the same as when considering OOV words (performance within 2%). However, when the OOV rate is high (for example as in Table 4, where the Greek data set has 22.77% OOV words), ignoring OOV words can noticeably damage performance of most smoothing techniques (up to a 7% reduction). This implies that considering OOV words is better than ignoring them when the sparse data problem is serious.

Effect of smoothing technique Another obvious phenomena is that the different smoothing techniques behave quite differently in our experiments. In

n	Absolute	Good-Turing	Linear	Witten-Bell
1	0.5700	0.5500	0.5500	0.5500
2	0.8500	0.8000	0.8400	0.8400
3	0.9000	0.7900	0.8900	0.8900
4	0.8700	0.7900	0.8500	0.8800
5	0.8600	0.7900	0.8700	0.8600
6	0.8600	0.7900	0.8700	0.8600

Table 7. Results on Greek authorship attribution with character models (only 0.12% OOV characters encountered)

n	Absolute	Good-Turing	Linear	Witten-Bell
1	0.2214	0.2211	0.2211	0.2211
2	0.6760	0.6861	0.6837	0.6686
3	0.8645	0.8600	0.8584	0.8629
4	0.8831	0.8757	0.8746	0.8865
5	0.8863	0.8717	0.8823	0.8897
6	0.8863	0.8791	0.8794	0.8908
7	0.8855	0.8783	0.8844	0.8900
8	0.8828	0.8730	0.8818	0.8892
9	0.8796	0.8743	0.8823	0.8897

Table 8. Topic detection results on English 20 Newsgroup data with character models (no OOV characters)

all cases, absolute smoothing and linear smoothing generally perform better than the other techniques. Although Good-Turing has been found to perform best in language modeling proper [2], it does not appear to perform as well in our text classification experiments. Also, Witten-Bell smoothing does not perform as well (on the word level model) despite the fact that it has been the benchmark smoothing technique in text compression [1] for more than a decade. We also find that Witten-Bell smoothing performs similarly to Laplace smoothing (corresponding to the similarity in their formulas), although Laplace smoothing works much worse in some cases (about 20% worse than absolute smoothing in the 1-gram case in Table 4). Overall, absolute and linear smoothing performed best, and Laplace smoothing performed worst in our experiments.

Effect of context length *n* Relaxing the naive Bayes independence assumption to consider local context dependencies is one of the main motivations of the CAN Bayes model. From Table 7 and Table 8, we can obviously see the increase of classification accuracy with longer context. However, the increase stops at some point (3 for the Greek data set and 5 for the English data set). This is due to the sparseness of data when larger n is used. That is, the additional information provided by longer context becomes compromised by data sparseness. Data sparseness also explains why the context information does not help in the Chinese data beyond 2-grams. The performance increase 3-4% from 1-gram to 2-gram, 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 1-2 characters. Bi-grams have been found very effective in Chinese IR [12]. Longer context information does not help 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), because the sparse data problem at the word level is more much 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.

Character level versus word level For many Asian languages such as Chinese and Japanese, where word segmentation is a hard, our character level CAN Bayes model is well suited for text classification because it avoids the need for word segmentation. For Western languages such as Greek and English, we can work at both the word and character levels. In our experiments, we actually found that the character level models worked slightly better than the word level models in the English 20 Newsgroup data set (89% vs. 88%). It appears that the character level models are capturing some regularity that the word level models are missing in this case. By relaxing the context, character level models can also capture regularities at the word level, and even phrase level regularities. However, in our experiments on the Greek data, we found that the character level models performed worse than the word level models (90% vs. 96%). So far, it remains inconclusive which level is the best to use for text classification with Western languages. This suggests that perhaps combining the two levels would result in more robust and better results.

Overall performance compared to state-of-the-art The results we have reported here are comparable to (or much better than) the start-of-the-art results on the same data sets. Our 96% accuracy on the Greek authorship attribution is much better than 70% reported in [26] which is based on a much more complicated analysis. Our 89.08% accuracy on the 20 Newsgroups data set is better than the best results 87.5% reported in [22] which is based on a combination of SVM and error correcting output coding (ECOC). Our 81.33% accuracy on the Chinese TREC data is comparable to the best results (about 82%) reported in [9] which is based on SVM, word segmentation, and feature selection. 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.

5.4 Relationship to Other Work

Using n-gram language models for text categorization has been considered in [3]. However, this work 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 feature selection at all. All n-grams remain in the model, and their importance is implicitly considered by their contribution to perplexity. Using language modeling techniques for text categorization is a new research area in IR and there is much work that can be done. Teahan and Harper [27] used a PPM compression method [1] for text categorization where they seek a model that obtains the best compression on a new document. The PPM compression model deals with OOV words with an escape method that uses Witten-Bell smoothing. Although Witten-Bell smoothing is best for compression, our experiments show it does not do best for text categorization. Work on considering context information for learning text classifiers include context sensitive rule learning and on-line learning [4]. 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 [8, 11, 29]. 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.

6 Conclusions 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 Markov chain. By using better smoothing techniques than Laplace smoothing we obtain further performance improvements. Our CAN Bayes modeling approach 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. Our experimental results support the utility of our approach.

We are investigating ways to combine word level and character level models to obtain better text classifiers. We are also investigating the issue of incorporating unlabeled data into training a CAN Bayes model using EM, similar to naive Bayes training [19].

Acknowledgments We thank E. Stamatatos for supplying us with the Greek authorship attribution data, and Ji He for the Chinese topic detection data.

References

1. T. Bell, J. Cleary and I. Witten. (1990). *Text Compression.* Prentice Hall.
2. S. Chen and J. Goodman. (1998). An Empirical Study of Smoothing Techniques for Language Modeling. *Technical report*, TR-10-98, Harvard University.
3. W. Cavnar, J. Trenkle. (1994). N-Gram-Based Text Categorization. In Proceedings of SDAIR-94.
4. W. Cohen and Y. Singer. (1999). Context-sensitive Learning Methods for Text Categorization, In *ACM Transactions on Information Systems*, v.17 pp 141-173.

5. P. Domingos and M. Pazzani. (1997). Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. *Machine Learning*, 29, 103-130
6. R. Duda and P. Hart. (1973). *Pattern Classification and Scene Analysis*. Wiley, NY.
7. S. Eyheramendy, D. Lewis and D. Madigan. (2003). On the Naive Bayes Model for Text Categorization. To appear in *Artificial Intelligence & Statistics 2003*.
8. N. Friedman, D. Geiger, and M. Goldszmidt. (1997). Bayesian Network Classifiers. In *Machine Learning* 29:131–163.
9. 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*.
10. D. Hiemstra. (2001). Using Language Models for Information Retrieval. Ph.D. Thesis, Centre for Telematics and Information Technology, University of Twente.
11. E. Keogh and M. Pazzanni. (1999). Learning Augmented Bayesian Classifiers: A Comparison of Distribution-based and Classification-based Approaches. In *Artificial Intelligence & Statistics 1999*
12. K. Kwok. (1999). Employing Multiple Representations for Chinese Information Retrieval, *JASIS*, 50(8), 709-723.
13. J. Lafferty and C. Zhai. (2001). Document Language Models, Query Models, and Risk Minimization for Information Retrieval. In *Proceedings SIGIR 2001*.
14. D. Lewis. (1998). Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In *Proceedings ECML-98*.
15. C. Manning, and H. Schütze. (1999). *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, Massachusetts.
16. A. McCallum and K. Nigam. (1998). A Comparison of Event Models for Naive Bayes Text Classification. In *Proceedings of AAAI-98 Workshop on "Learning for Text Categorization"*, AAAI Presss.
17. D., Mladenic. (1998). Feature subset selection in text-learning. In *Proceedings of ECML98*.
18. H. Ney, U. Essen, and R. Kneser. (1994). On Structuring Probabilistic Dependencies in Stochastic Language Modeling. In *Comput. Speech and Lang.*, 8(1), 1-28.
19. K. Nigam, et al. (2000). Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2/3). pp. 103-134.
20. M. Pazzani and D. Billsus. (1997). Learning and Revising User Profiles: The identification of interesting web sites. *Machine Learning*, 27, 313-331.
21. J. Ponte, W. Croft. (1998). A Language Modeling Approach to Information Retrieval. In *Proceedings of SIGIR1998*, 275-281.
22. J. Rennie. (2001). Improving Multi-class Text Classification with Naive Bayes. *Master's Thesis*. M.I.T. AI Technical Report AITR-2001-004. 2001.
23. I. Rish. (2001). An Empirical Study of the Naive Bayes Classifier. In *Proceedings of IJCAI-01 Workshop on Empirical Methods in Artificial Intelligence*.
24. S. Scott and S. Matwin. (1999). Feature Engineering for Text Classification. In *Proceedings of ICML' 99*, pp. 379-388.
25. F. Sebastiani. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1-47, 2002.
26. E. Stamatatos, N. Fakotakis and G. Kokkinakis. (2000). Automatic Text Categorization in Terms of Genre and Author. *Comput. Ling.*, 26(4), pp.471-495.
27. W. Teahan and D. Harper. (2001). Using Compression-Based Language Models for Text Categorization. In *Proceedings of Workshop on LMIR*.
28. Y. Yang. (1999). An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, Vol. 1, No. 1/2, pp 67–88.
29. H. Zhang and C. Ling. (2001). Learnability of Augmented Naive Bayes in Nominal Domains. In *Proceedings of ICML2001*.