UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

TEXT-BASED LANGUAGE IDENTIFICATION FOR THE SOUTH AFRICAN LANGUAGES

GERRIT REINIER BOTHA

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

TEXT-BASED LANGUAGE IDENTIFICATION FOR THE SOUTH AFRICAN LANGUAGES

by

Gerrit Reinier Botha

Submitted in partial fulfillment of the requirements for the degree

Master of Engineering (Electronic Engineering)

in the

Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

Supervisor: Professor E. Barnard

February 2008

SUMMARY

TEXT-BASED LANGUAGE IDENTIFICATION FOR
THE SOUTH AFRICAN LANGUAGES

by

Gerrit Reinier Botha

Supervisor: Professor E. Barnard

Department of Electrical, Electronic, and Computer Engineering

Master of Engineering (Electronic)

We investigate the factors that determine the performance of text-based language identification, with a particular focus on the 11 official languages of South Africa. Our study uses n-gram statistics as features for classification. In particular, we compare support vector machines, Naïve Bayesian and difference-in-frequency classifiers on different amounts of input text and various values of n, for different amounts of training data. For a fixed value of n the support vector machines generally outperforms the other classifiers, but the simpler classifiers are able to handle larger values of n. The additional computational complexity of training the support vector machine classifier may not be justified in light of importance of using a large value of n, except possibly for small sizes of the input window when limited training data is available.

We find that it is more difficult to discriminate languages within language families then those across families. The accuracy on small input strings is low due to this reason, but for input strings of 100 characters or more there is only a slight confusion within families and accuracies as high as 99.4% are achieved. For the smallest input strings studied here, which consist of 15 characters, the best accuracy achieved is only 83%, but when the languages in different families are grouped together, this corresponds to a usable 95.1% accuracy.

The relationship between the amount of training data and the accuracy achieved is found to depend on the window size for the largest window (300 characters) about 400 000 characters are sufficient to achieve close-to-optimal accuracy, whereas improvements in

accuracy are found even beyond 1.6 million characters of training data.

Finally, we show that the confusions between the different languages in our set can be used to derive informative graphical representations of the relationships between the languages.

Keywords: text-based language identification, *n*-gram statistics, Naïve Bayesian classification, difference-in-frequency classification, support vector machine

OPSOMMING

TEKS-GEBASEERDE TAALHERKENNING VIR
DIE SUID-AFRIKAANSE TALE

deur

Gerrit Reinier Botha

Promoter: Professor E. Barnard

Departement Elektries, Elektronies en Rekenaar-Ingenieurswese

Meesters in Ingenieurswese (Elektronies)

Ons ondersoek die faktore wat die verrigting van teks-gebaseerde taalherkenning bepaal, met die klem op die 11 amptelike tale van Suid-Afrika. Ons navorsing maak gebruik van *n*-gram statistieke as kenmerke. Ons vergelyk drie klassifiseerders: ondersteuningsvektormasjiene, Naïewe Bayes klassifiseerders en verskil-in-frekwensie klassifiseerders, vir verskillende hoeveelhede invoerteks met wisselende waardes vir *n*, vir verskeie hoeveelhede afrig data. Vir 'n vasgestelde waarde van *n* presteer die ondersteuningsvektormasjien meestal beter as die ander klassifiseerders, maar die eenvoudiger klassifiseerders kan met groter waardes van *n* werk. Die beperking op *n* en die berekeningskoste van die ondersteuningsvektormasjien regverdig nie die gebruik daarvan nie, behalwe waar die invoervenstergrootte klein is en min afrig data beskikbaar is.

Ons bevind dit moeiliker om te onderskei tussen tale binne 'n taalfamilie as tussen die van verskillende families. As gevolg hiervan is die akuraatheid vir klein invoerstringe baie laag, maar vir die invoer van 100 of meer karakters is daar min verwarring binne families en akkuraatheid van so hoog as 99.4% is behaal. Vir die kleinste invoerstring van 15 karakters is die hoogste akkuraatheid slegs 83%, maar wanneer die tale in die verskillende families gegroepeer word, vermeerder die akkuraatheid na 95.1%.

Dit is bevind dat die verhouding tussen die grootte van die afrigdata en die akkuraatheid afhanklik is van die venstergrootte. Vir die grootste venster van 300 karakters is ongeveer 400 000 karakters genoeg om omtrent optimale akkuraatheid te behaal. Vir ander vensters verbeter die akkuraatheid selfs vir meer as 1.6 miljoen karakters.

Laastens toon ons dat verwarrings tussen die verskillende tale gebruik kan word om ’n visuele intrepretasie van die verhouding tussen tale aan te dui.

Sleutelwoorde: taalherkenning gebaseer op teks, *n*-gram statistieke, Naïewe Bayesian klasifiseerder, verskil-in-frekwensie klasifiseerder, ondersteuningsvektormasjien (Engels: support vector machine)

LIST OF ABBREVIATIONS

IDF	Inverse Document Frequency
HTML	Hypertext Markup Language
LID	Language Identification
MDS	Multi-dimensional Scaling
NB	Naïve Bayesian
OGI	Oregon Graduate Institute
RBF	Radial Basis Function
SMS	Short Message Service
SVM	Support Vector Machine
UTF	Unicode Transformation Code

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION

1.1 LANGUAGE PROCESSING IN A MULTILINGUAL ENVIRONMENT

In a multilingual environment, language processing is often initiated with some form of language identification (LID). For example:

- A document processing system needs to determine the language of textual contents before topic identification, translation, stemming or search.
- A text-to-speech system may need to determine the source language of short pieces of text to determine pronunciation rules, prosodic models and phrasing strategies.
- A general-purpose telephonic help line may require LID in order to route calls to an appropriate operator [1].

Automatic speech-based LID approaches identify the language based on features such as the probabilities of the phoneme sequence extracted from the acoustic signal [2]. In LID from text, statistical features in the text or linguistic knowledge are used in classification.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

The aim of this dissertation is to investigate text-based LID and to understand the factors that determine how accurately this task can be accomplished. We base our investigation on the 11 official languages of South Africa. We compare 3 different classifiers and investigate factors which influence classification accuracy. The classifiers are trained and tested using n-gram statistics computed from sets of text data.

1.2 PROBLEM STATEMENT

Although text-based LID has been studied extensively, there is still not a general understanding of the factors that determine classification accuracy. Such factors include the following:

- *The size of the textual fragment to identify* - the less feature statistics are available, the more difficult it is to distinguish between languages. In many applications it is desirable to identify the language from a limited amount of text. For example, when code switching is likely to occur, one would like to identify languages based on only a few words; e.g. for Short Messages (SMSs), ten to thirty words are typically available.

- *The amount and variety of training data available* - in the major world languages, corpora measured in millions of words are not uncommon, but in most of the languages of the world, algorithm development has to proceed from much less text. The domain from which the training text is extracted is potentially important in that context since limited training data will lack the full variety of a language and is therefore, expected to generalize less successfully to new domains.

- *The classification algorithm employed* - the number of occurrences of each n-gram in a text to be identified can be thought of as the components of a vector and numerous algorithms are potentially applicable to the classification of such vectors. However, the number of possible n-grams grows exponentially with n, which restricts the set of applicable algorithms to those that can handle very large feature vectors.

The respective effects of these various factors on classification accuracy, and their interaction, will form the core of the current research study.

In addition, distinguishing languages belonging to the same phylogenetic families is

much harder than identifying languages that fall outside such families. The 11 official languages of South Africa can be grouped into a number of language families and sub-families: Nguni (consisting of, isiZulu, isiXhosa, isiNdebele and Siswati), Sotho (consisting of Sesotho, Sesotho Sa Leboa and Setswana), Germanic (English and Afrikaans) and a pair that falls outside these families (Xitsonga and Tshivenda).

Text-based LID has mainly been studied for the major languages of the world. The only other research (to our knowledge) that includes the South African languages in a text-based LID task was done by Combrinck and Botha [3]. They reported a substantially lower performance rate for South African languages in comparison to a set of European languages. Unfortunately, they do not report error rates, and do not provide sufficient information to provide an understanding of the level of success that can be expected with these languages. Thus text-based LID for the South African languages is still open for research.

1.3 CONTRIBUTION

The primary contribution of the current research is a detailed analysis of the interaction of the various factors mentioned in Section 1.2, This analysis makes it possible to provide general guidelines on the development of a text-based LID system for any set of languages; it also provides detailed information on the accuracy that can be achieved with the particular set of languages spoken in South Africa.

The complete datasets, findings and results developed in this research are publicly available. The available datasets can be used in further research studies. There is currently no other common dataset that is widely used for text-based LID. This makes a difficult to compare results in different reported studies because classifiers were trained from different sources.

The complete datasets, findings and results developed in this research are publicly available. The available datasets can be used in further research studies. There is currently

no other common dataset that is widely used for text-based LID. This makes a difficult to compare results in different reported studies because classifiers were trained from different sources.

A software module is created for possible integration in language processing systems. Using our best classifier to classify between 11 South African languages, an accuracy of at least 98.5% is achieved when classifying strings of more then a 100 characters.

1.4 OBJECTIVE

The central aim of the study is to investigate the accuracy that can be achieved for the South African languages using *n*-gram statistics. Factors that can influence the accuracy will be investigated, such as:

- The size of the textual fragment used.
- The amount and variety of training data.
- The classification algorithm employed.
- The similarity of languages to be distinguished.

1.5 THESIS OVERVIEW

The layout of the dissertation is as follow:

- Chapter 2 covers the background of text-based LID, techniques and current contributions to the field.
- Chapter 3 discusses the implementation of the classifiers.
- Chapter 4 presents our experimental setup and results.
- Chapter 5 is a summary of our research and a discussion on possible future work and applications.

CHAPTER TWO

BACKGROUND

2.1 INTRODUCTION

Automatic LID is an integral part in many multilingual language processing systems, as described in Chapter 1. LID can be divided into two classes: spoken and written LID. Spoken LID methods make use of signal processing techniques where LID from text is a symbolic processing task.

Spoken LID is considered a more difficult task due to the problem of variability in speech. A thorough review of spoken LID is provided in [2], and more recent techniques are covered in [4]. Although Although text-based classification is sufficiently advanced to be in widespread commercial use, many factors that influence this task are not well understood; therefore, improvements can still be made. In the remainder of this dissertation, our focus is on text-based LID.

The chapter provides background on text-based LID and techniques. The chapter layout is as follows:

- Section 2.2 is an overview of LID from text.
- In Sections 2.3 and 2.4 we discuss text-based LID approaches.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

2.2 LANGUAGE IDENTIFICATION FROM WRITTEN TEXT

The general topic of text-based LID has been studied extensively, and a spectrum of approaches has been proposed with the most important distinguishing factor being the depth of linguistic processing that is utilized. At the one extreme of complexity are approaches that attempt to do a complete parse of text in order to determine not only the language used, but also the syntactic structure of the textual fragment. These linguistic models are (by definition) perfectly accurate. However, they require substantial resources for their development and can be computationally expensive if a large set of languages has to be considered.

The opposite extreme of complexity attempts to identify the language by using simple statistical measures of the text under consideration. For example, statistics can be gathered from:

- letter sequences
- presence of certain keywords
- frequencies of short words
- unique or highly distinctive letters or short character strings

Conventional algorithms from pattern recognition are then used to perform text-based LID based on these statistics. *N*-gram statistics is a well known choice for building statistical models and we discuss this feature in the next section.

2.3 *N*-GRAMS

An *n*-gram is a sequence of *n* consecutive letters. The *n*-grams of a string are gathered by extracting adjacent groups of *n* letters. The *n*-gram combinations for the string “example” is shown in Figure 2.1.

bi–grams	**:**	**ex**	**xa**	**am**	**mp**	**pl**	**le**
tri–grams	**:**	**exa**	**xam**	**amp**	**mpl**	**ple**	
quad–grams	**:**	**exam**	**xamp**	**ampl**	**mple**		

Figure 2.1: *N*-gram combinations for the string “example”.

In *n*-gram based methods for text-based LID, frequency statistics of *n*-gram occurrences are used as features in classification. The advantage is that no linguistic knowledge needs to be gathered to construct a classifier. The *n*-grams are also extremely simple to compute for any given text, which allows a straightforward trade-off between accuracy and complexity (through the adjustment of *n*) and have been shown to perform well in text-based LID and related tasks in several languages. Increasing the size of *n* can increase the accuracy of the classifier (since a larger window of characters is considered), but beyond a certain level the large number of possible *n*-grams is too sparsely represented in any given corpus, and accuracy decreases thereafter.

The number of possible *n*-gram combinations depends on:

- the value of *n*
- the number of distinct “characters” contained in the orthography employed

The size of the feature space grows exponentially with *n* and is given by a^n, where a is the size of the distinct character set.

An increase in the number of *n*-gram combinations increases the complexity of the training model which results in long training times and extensive resource usage. Some classifiers are affected more than others and we will see that this is an important factor in classifier selection.

A larger character string will obviously contain more *n*-grams and more confident

statistical measures can be calculated. The number of *n*-grams in a character string is equal to $\alpha - n + 1$, where α is the length of the character string.

2.4 TEXT-BASED LANGUAGE IDENTIFICATION APPROACHES

2.4.1 *N*-GRAM RANK ORDERING

N-gram rank ordering was introduced by Cavnar and Trenkle [5]. In this approach, *N*-gram combinations in the training and testing sets are ordered from most to least frequent. The rank difference of each *n*-gram in the testing and the corresponding rank in the training model is calculated. The sum of all the rank distances is then calculated and the testing language with the smallest distance is identified as the most probable language. If an *n*-gram is not present in the training set, a maximum ranking distance is given. Figure 2.2 illustrates the ranking calculation for a test string.

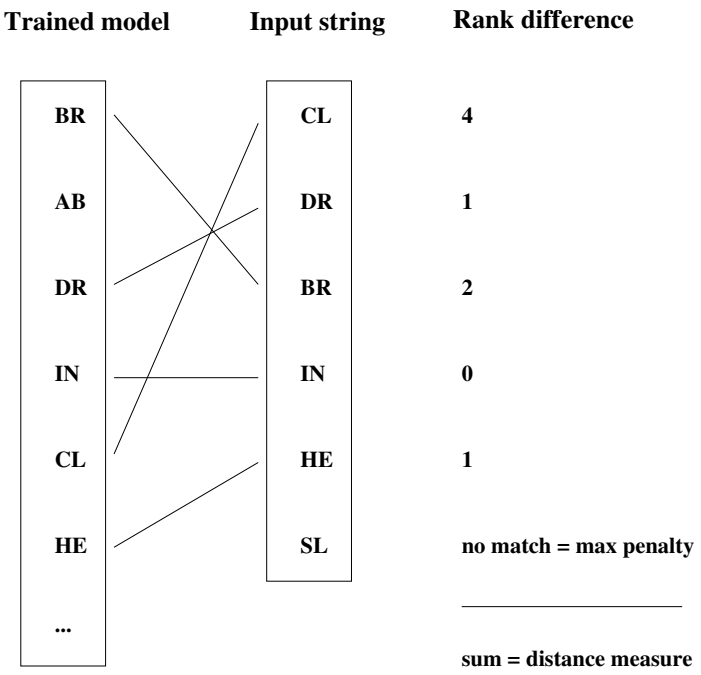

Figure 2.2: An *n*-gram rank ordering example. The figure illustrates the process of calculating the distant measure.

Cavnar and Trenkle used the technique to test LID on newsgroup text in 8 languages. These languages were English, Polish, Dutch and German (Germanic group), Portuguese,

French, Spanish and Italian (Romance Group). Training sets for the languages varied between 20K and 120K characters. Using the 300 most frequent 3-grams the best accuracy of 98.6% was achieved for a test string smaller than 300 characters. From the results it is difficult to conclude what the performance on smaller test strings (e.g. $<$ 20 characters) would be.

2.4.2 NAÏVE BAYESIAN CLASSIFIER

The Naïve Bayesian (NB) method seeks to maximize the probability of a language given a document $P(L|D)$, where L is a language, and D is a document. Using Bayes rule, we rewrite this as

$$\max P(L|D) = \max \frac{P(L)P(D|L)}{P(D)}. \tag{2.1}$$

Since the denominator is the same for all languages, and since all languages are assumed to be equally probable, we need only to maximize $P(D|L)$. Equation 2.1 can therefore be rewritten as

$$\max P(L|D) = \max P(D|L). \tag{2.2}$$

By calculating the maximum probability of a document given a language, we calculate the most probable language given that document. If *n*-gram statistics are used in the classification model, it is assumed that successive *n*-grams are independent of one another, so that the various log likelihoods can be added together.

By utilizing this approach, Dunning [6] estimates the likelihood of a string belongs to a language model. Experiments were done on English and Spanish parallel texts. An accuracy rate of 92% was obtained with 50K characters of training and a 20 character string. If the test string size was increased to 500 characters the accuracy increased to 99%. For 5K characters of training and a 500 character test string the accuracy dropped to 97%. The evaluated languages were from different language families (Romance and Germanic) and therefore it is difficult to compare results with classification done on languages from the same family groups.

Kikui [7] proposed an algorithm for simultaneously identifying the coding system and the language of a given string. The coding system was identified heuristically, where the LID part was based on the NB model. Documents were collected from the Internet and grouped into 700 training and 640 testing documents. The tests included 9 languages (English, German, Spanish, French, Italian, Portuguese, Japanese, Chinese and Korean) and 11 coding schemes. They found that *n*=4 gave the lowest error rates and that their accuracies were equivalent to the results presented in [5].

Beesley [8] calculated the likelihood of each word in a sentence using bi-gram statistics. The language was identified on the overall score of each word. We believe that *n*-grams between word boundaries still hold valuable information to discriminate between languages.

Grefenstette [9] compares two feature statistics in building a likelihood model. The one statistical model contains the likelihood of all tri-grams that appeared more than 100 times in a training set. The other model contains the likelihood of all words which are smaller than 5 characters and occur more than 3 times in the training text. It is found that for sentences smaller than 15 words the tri-gram method performs best but for a greater number of words both methods perform well. A 15 word window can roughly approximate a 100 characters. Therefore it's difficult to conclude what the error rate for much smaller test strings would be.

2.4.3 CLASSIFICATION BASED ON DOT PRODUCT CALCULATION

Let the vector $\mathbf{c}$ represent all the *n*-gram combinations. The dot-product method builds a language vector $\mathbf{l}_j$ from the *n*-gram statistics of a document, where l_{ij} is the frequency of the *n*-gram c_i. To classify a test string, a vector $\mathbf{x}$ is built from the n-gram statistics of the test string. The normalized dot-product between a language vector $\mathbf{l}_j$ and test string $\mathbf{x}$ is then computed by

$$\mathbf{x} \cdot \mathbf{l}_j = \frac{\sum_{i=1}^{N} x_i l_{ij}}{|\mathbf{x}||\mathbf{l}_j|}, \tag{2.3}$$

where N is the total number of n-gram combinations.

The computed measure indicates how close the two vectors are and therefore how close the text string is to the language model. The closer the measurement is to 1, the more similar the vectors become. Thus, the language model with the highest measurement will be identified as the language of the text segment.

Damashek [10] implemented the method to measure similarities between documents in 31 languages. Using *5*-gram statistics the languages could be accurately grouped by family.

In [11], Prager gives each vector item an inverse document frequency (IDF) weighting. The IDF is the inverse of the number of languages that contain the n-gram combination. For closely-related languages Prager found that this weighting stayed fairly fixed. By combining *4*-gram and word statistics (with no restriction on the word length), best performance was achieved. Thirteen West-European languages were evaluated. The average performance was 85.4% for a 20 character input string and over 99% for a 130 characters and up.

Padro and Padro [12] compared the NB method, the dot-product classification and the n-gram rank ordering method to each other. Identification was tested on 6 languages (English, Catalan, Spanish, Italian, German and Dutch). Overall, the NB classifier proved best (significantly so for small test samples), followed by the dot-product classifier and then n-gram rank ordering method.

2.4.4 CENTROID BASED

Kruenkrai et al. [13] implemented a classifier closely related to the classifier implementation in [5]. The language profile is a vector $\mathbf{l}_j$ containing the likelihoods of n-grams in the training document(s). The testing vector $\mathbf{x}$, is built from likelihoods of the n-grams in the test string. A classification decision is done based on the squared Euclidian distance, given by

$$D_{CB} = ||\mathbf{l}_j - \mathbf{x}||^2. \tag{2.4}$$

The distance indicates similarity of the test string to the language profile. The profile that yields the smallest distance will be classified as the language of the string. The results of their experiments were compared with a support vector machine (SVM) and will be discussed in Section 2.4.6.

2.4.5 CLASSIFICATION USING THE RELATIVE ENTROPY

Relative entropy (also known as the Kullback Leibler distance) is used in information theory to measure the difference between two probability distributions. For a discrete random variable x, and probability distributions p and q, the relative entropy is given by

$$D_{KL} = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}. \tag{2.5}$$

In [14] the p distribution was associated with the test set, the q distribution was associated with the training set and X is the samples in distribution p. If an *n*-gram is not found in the trained model, the equation above is undefined; therefore a penalty of 0.5 is added for such events.

The language of the test string is assigned to the language profile with the smallest relative entropy. Data was obtained from the European Corpus Initiatives CD-ROM and a subset of documents in 18 Roman alphabet languages were used for training and testing purposes. Classification was compared by training the classifier with uni-gram and bi-gram statistics. The average accuracy for a 200 line training set and 1 test line is 78.6% for uni-grams and 90.2% for bi-grams. If the number of test lines is increased to 10, the uni-gram accuracy increases to 98.5% and the bi-gram accuracy to 99.9%. The accuracy for a 2000 line training set and 1 test line is 81.5% for uni-grams and 94.1% for bi-grams. For 10 lines the uni-gram accuracy increases to 99.0% and the bi-gram accuracy to 99.9%.

2.4.6 SUPPORT VECTOR MACHINES

The SVM is a non-linear discriminant function that is able to generalize well, even in high-dimensional spaces (a full description of the classifier can be found in Appendix A). The classifier maps input vectors to a higher dimensional space where a separating hyper-plane is constructed. The hyper-plane maximizes the margin between the two datasets [15]. In real-world problems data can be noisy and the classifier would usually overfit the data. For such data, constrains on the classifiers is relaxed by introducing slack variables. This improves overall generalization [16].

Kruengkrai et al. [13] compared the performance of two kernel classifiers: the centroid method (Section 2.4.4) and an SVM. A string kernel was implemented which computes the inner product in the feature space generated by all subsequences of length k. A subsequence can be of any combination of k characters. The difference between *n*-grams and these combinations is that the subsequences do not need to follow contiguously. The subsequences are weighed by an exponential decaying factor, thus the subsequences that follow contiguously weighs more. A subsequence of k=5 and a decay factor of 1 gave best results. An n-gram rank ordering classification system was used as baseline in comparisons. Performance was tested based on 5-fold cross validation on 17 languages and an average of 578 sentences per language. On a test sample (with an average length of 50 character per item) n-gram rank ordering performed with 90.2% accuracy, the centroid method with 95.9% accuracy and the SVM gave an overall best performance accuracy of 99.7%

Zhai et al. [17] found that a reduction in the feature space of the SVM results in a significant decrease in performance accuracy. They also concluded that the SVM is highly sensitive to prior distributions. This is due to the nature of the SVM classifier that does not compensate for different sizes of training data. Therefore classification is biased towards the class with the larger training set.

In [18], Lodhi et al. showed that an SVM trained with *n*-gram statistics outperforms an SVM using kernel strings as features. Though it would be assumed that a kernel string would capture more information of a language, it probably introduces complexity into the

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

SVM, which has a negative impact on decision-making.

2.4.7 CLASSIFICATION BASED ON CONFIDENCE INTERVALS

Elworthy [19] studied the possibilities of using confidence intervals in text-based LID. In any point of the decision-making process, the algorithm terminates when a confident decision can be made. Probabilities are calculated by assuming a binomial distribution. A decision is made when the probability of the lower confidence limit of the current best language profile exceeds that of the probability of the upper confidence limit of the second best profile. The reason behind this approach is to provide feedback on the classification process.

Poutsma [20] worked on a similar concept. Assuming a binomial distribution, Monte Carlo sampling was used in calculating probabilities. Whenever a certain confidence limit was reached a classification decision was made. The technique was compared with a rank ordering algorithm and it performed with less accurate results.

2.4.8 MARKOV MODELS

Xafopoulos et al. [21] propose a system that is based on hidden Markov models (HMM) to enable character sequence modeling. Identification performance was tested on 5 European languages (English, German, French, Spanish and Italian). The system accuracy was 1% less than the rank ordering method [5] for test sentences between 20 and 100 characters.

House and Neuberg [22] reduced a string of text to a four character alphabet. These character sequences formed finite-state models for a language profile. Though the research was a study on spoken LID, the method is also applicable to text-based identification.

Binas [23] applied three Markov models for text-based LID: the simple Markov model, the aggregate Markov model (AGMM) and the hidden Markov model (HMM). Their results show that the AGMM and HMM slightly outperform the simple Markov model which is equivalent to a NB classifier.

2.4.9 DECISION TREES

Hakkinen and Tian [24] compared a NB classifier with a classifier based on a decision tree. *N*-grams statistics were enhanced by dividing a word into body, head and tail and training the classifiers on the *n*-gram statistics of these parts. To compute the likelihood, the *n*-gram is chosen according to its position in the input letter sequence. The reason for creating the enhanced *n*-grams was to incorporate distinct linguistic characteristics such as the prefixes and suffixes in a language. The decision tree makes a classification decision by asking a series of questions about the context of each letter in the input string. Performance accuracy was tested on short names in 4 languages (English, Finnish, Spanish and German). The decision tree achieved an accuracy of 66% where the tri-gram trained NB classifier performed with 71.8% accuracy.

2.4.10 NEURAL NETWORKS

Macnamara et al. [25] compared a NB classifier with a neural net classifier. For the NB classifier the 100 most discriminative *n*-grams and the 100 most frequent *n*-grams were extracted from the training text. An input string was first scanned for discriminant *n*-grams and a decision was made once one was found. If no discriminant *n*-grams occurred the log likelihood was calculated using probabilities of the 100 most frequent *n*-grams and the language with the highest score was classified as the language of the input string. Tests were performed on 18 languages. For an average of 2 to 3 words the accuracy of the neural network was 70% and for the tri-gram trained NB classifier an accuracy of 83% was found.

In [26], Tian and Suontausta proposed a neural network for written text LID where the memory resources are sparse.

2.4.11 MULTIPLE LINEAR REGRESSION

Murthy and Kumar [27] implemented a multiple linear regression model to discriminate between pairs of Indian languages. Aksharas, which are smaller units than words (somewhat similar to syllables), are used as feature statistics. Experimental results are comparable with published results based on identification of European languages.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

2.4.12 OTHER APPROACHES

Jalam and Teytaud [28] tested a new kernel method based on a dissimilarity measure. LID was tested on 5 languages (French, Arabic, English, Spanish and German). For each language 25KB of text was used to train the models. Three classifiers were compared: a radial basis function (RBF) with a Gaussian kernel, a neural network using Kullback Leibler dissimilarity measurements and a neural network using their own kernel method. On a testing set of 100 character samples, the RBF showed 99.2% accuracy, the neural network using Kullback Leibler dissimilarity measurements performed with 99.7% accuracy and the neural network using the kernel method performed with 99.2% accuracy. For 20 character samples the RBF accuracy dropped to 71% accuracy, the neural network using Kullback Leibler dissimilarity measurements dropped to 47.2% accuracy and the neural network using the kernel method dropped to 88.4% accuracy.

Souter et al. [29] compared three LID approaches on a set of 9 languages. The first method searched for unique short character strings. The second method builds a model from the 100 most frequent words in a language. The language profile in which the most words of an input string occur is chosen as the most likely language. The third method calculates probabilities from tri-gram statistics using the NB method. The NB method gave the best performance overall, needing 25%-50% of the n-gram combinations occurring in the language profile to achieve optimal recognition.

Giguet [30] implemented a method that combines linguistic knowledge and statistics. The method first scans for unique strings. If a decision cannot be made, the method determines if a word belongs to the list of grammatical words. If the previous step was indecisive, n-gram probabilities are used to make a final decision. The advantage of the approach is that the grammatical word method and the n-gram statistics complement each other so that identification accuracy can be improved.

Linsand and Goncalves [31] present a purely linguistic approach to written text LID which is based on closed grammatical classes. By including more languages in the algorithm more accurate classification is obtained, but this method clearly requires significant

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

linguistic expertise in each target language.

The only published results of text-based LID for South African languages are presented in [3]. The language model contains the most distinctive tri-grams of the language. For each language, a count is kept for the number of distinctive tri-gram hits in the test string. This count is then normalized by the total number of characters in the string. The language model with the highest score is classified as the language of the test string. The disadvantage of this model is that each language model is dependent on the other language models and is not uniquely defined.

2.5 CONCLUSION

Written text LID can be approached from a pattern recognition perspective by considering statistical characteristics in text as feature measures. Statistical measures can be based on frequency of words, unique letters, short strings or keywords found in a document. *N*-grams are easy to compute and efficient in text-based LID.

The study of text-based LID is a well-known topic and a multitude of approaches have been proposed. The simple and effective methods include:

- The NB calculates the likelihood of consecutive n-grams by assuming no correlation between *n*-gram combinations.
- Measures as the dot-product, Euclidean distance, relative entropy or *n*-gram rank ordering which indicates the similarity between a language model and that of an input vector.

Decision trees, hidden Markov models, neural networks and SVMs are approaches from a more conventional pattern recognition background. Though it may be expected that these classifiers would prove more accurate in the task, published results demonstrate that it is still difficult to outperform the simpler methods. The accuracy of the simpler methods can be improved by increasing the feature dimensionality. In these high feature dimensions the more conventional methods become too complex to consider.

CHAPTER THREE

IMPLEMENTATION

3.1 INTRODUCTION

In Chapter 2 we discussed various methods that can be considered in text-based LID. In the studies, classifiers were evaluated differently by considering different conditions. In this chapter we present the conditions that we will use to evaluate the performances of our text-based LID.

The layout is as follow:

- Section 3.2 discusses the experimental approach that was followed for our experiments.
- Section 3.3 discusses the process of collecting and processing our text data.
- Section 3.4 gives more detail about our classification features.
- Section 3.5 presents our classification methods.

3.2 EXPERIMENTAL APPROACH

3.2.1 FEATURE SET

From the literature study (Chapter 2) it is obvious that many approaches can be followed in a text-based LID. A pure linguistic approach would be the best candidate where high classification accuracies are desired. Though these models would describe the language best, a large amount of linguistic expertise is required. Where such knowledge is not available (or the effort to gather such linguistic information is not justified) statistical approaches are a viable alternative. Statistical language models can be built from the statistics of words, letters or *n*-grams. The *n*-gram based models outperform a word based model for small text fragments and do equally well for larger fragments [9]. In another study *n*-grams achieved better results than string kernels [18]. This is also by far the most popular choice in the literature; we have therefore restricted our feature sets to *n*-gram based features.

3.2.2 CLASSIFIERS

In light of the large number of classifiers that have been applied to text-based LID, it was not feasible to experiment with all possible combinations. The classification algorithms employed, were similarly chosen for their proven performance in published studies, as well as their ability to clarify theoretical issues, as we discuss below.

The NB classifier was employed in many of the studies. The classifier achieves good results in various experiments. In only one study was this classifier slightly outperformed [23]. We therefore consider this classifier as a good baseline system for comparison with the other implemented classifiers.

The *n*-gram rank ordering method [5], is an approach which is referenced, used and compared in many studies. The method is mostly outperformed in comparison tests [12] [13]. We have decided to implement a somewhat similar classifier for comparative purposes. Like the dot-product, relative entropy and centroid based classifiers, this method also classifies based on a similarity measure between vectors. In our study we call this a difference-in-frequency classifier.

We employed the SVM from the class of more complex classifiers. The SVM generalizes well in high- dimensional spaces and showed good results in comparison tests for text-based LID [13], and has also performed competitively in many other pattern-recognition tasks. The classifier was not implemented from first principles; an available software module with full SVM functionality was used.

3.2.3 TRAINING AND TEST DATA

In published studies on text-based LID, a variety of methods have been used to compute classification accuracies. Classifiers were trained with different amounts of textual training data; in some, documents were limited to one domain, and others would span over a few domains. The metric for the size of the test string varied from number of characters, number of words, size in bytes (which will depend on the encoding scheme used), lines and sentences. Some tests were performed on languages without any family relationships and others within language families. Some evaluated performance accuracy using only a validation set where others performed a thorough cross-validation test. In studies where classifiers were compared against each other more reliable conclusions can be made, since the classifiers were evaluated under the same conditions. However, the comparison of results between different studies is generally impossible.

Our study focuses on the factors that can affect classification accuracy. Though our evaluation process uses some previous ideas, it is designed to make comparisons more reliable. All classifiers were evaluated under the same well-documented circumstances (training data, test data and character window) to assure reliable results.

3.2.4 MEASURING TEXT FRAGMENTS

Table 3.1 shows the average number of words, for each language, in a 2M character file. The table also shows the average number of characters for a word in each language. The most significant difference occurs between the Nguni and Sotho languages, which is not surprising given that the former are written conjunctively and the latter disjunctively. Thus, if training and testing would be performed on a window containing a specified number of

words, more *n*-gram statistics could be gathered for the Nguni languages. A larger string would have more n-grams which would lead to a better model and (artificially) better classification. If the number of bytes in a window are specified, a similar (though somewhat less severe) problem would be introduced, since different coding schemes such as UTF (Unicode Transformation Code) use different byte sizes to represent a character. Thus, a language set with multi-byte encoding would be influenced negatively. Character windows seemed the most reliable measure of window size (though disjunctively written agglutinative languages are again somewhat disadvantaged, because of the additional white space between word), and were used in our experiments.

Language	**Number of words**	**Average word length**	**15 chars**	**100 chars**	**300 chars**
Sesotho	397 891	5.03	2.98	19.89	59.68
Sesotho Sa Leboa	395 022	5.06	2.96	19.75	59.25
Setswana	384 237	5.21	2.88	19.21	57.64
isiXhosa	249 200	8.03	1.87	12.46	37.38
isiZulu	238 110	8.4	1.79	11.91	35.72
isiNdebele	228 977	8.73	1.72	11.45	34.35
Siswati	222 616	8.98	1.67	11.13	33.39
Tshivenda	377 905	5.29	2.83	18.9	56.69
Xitsonga	368 858	5.42	2.77	18.44	55.33
Afrikaans	335 950	5.95	2.52	16.8	50.39
English	373 057	5.36	2.8	18.65	55.96

Table 3.1: Word statistics on a 2M character file of each language. Average word lengths and number of words per character window are indicated.

3.2.4.1 DATASETS AND SAMPLE SIZES

Accuracy was evaluated for various amounts of training text. In the 10-fold cross validation we performed tests on 200K, 400K, 800K, 1.6M and 2M characters.

Three different character windows were used. Table 3.1 indicates the average number of words per character window. The choice of these sizes influences the difficulty of correct classification. A 15 character window represents 2-3 words, and is expected to be challenging for statistical methods. For a 100 character window (a long sentence) classification accuracy improves and at a 300 character window (paragraph) classification

will hopefully be highly accurate.

3.3 TEXT DATA

Texts from various domains in all 11 South African languages were obtained from Professor D.J. Prinsloo of the University of Pretoria. The data included text from various sources (such as newspapers, periodicals, books, the Bible and government documents) and therefore, the corpus spans several domains. Per language the size of text varied from 5MB to 6MB, except Afrikaans for which only 900KB was obtained. Extra Afrikaans text, to equal the amount of the other languages, was collected using a web crawler [32]. All text was encoded in the UTF-8 format to support special characters found in Afrikaans (è, é, ê, ë, ï, ò, ó, ô, ö, ú, û, and ü) and the 'š' in Sesotho Sa Leboa and Setswana.

Due to the diversity of sources employed, text was not homogeneous and needed some automatic preprocessing in order to be used for building models. For example, consecutive white spaces were replaced with a single space character. Moreover, numbers, names, abbreviations, punctuation marks and addresses (e.g. e-mail and links to Internet websites) were removed. After this preprocessing the size of the text was significantly reduced to 2MB to 3MB per language.

3.4 CLASSIFICATION FEATURES

For either a fixed-length sample or an unbounded amount of text, the frequency counts of all n-grams were calculated. The characters that can be included in n-gram combinations were a space, the 26 letters of the Roman alphabet, the other 14 special characters found in Afrikaans, Sesotho Sa Leboa and Setswana, and the unique combination ''n' , which functions as a single character in Afrikaans. No distinction was made between upper and lower case characters. In total the size of the character set added up to 42, which results in 42^n possible n-grams. However, many of these combinations are not present in any of the languages (for example the 3-gram aaa) and therefore the feature space is smaller than this number. No experiments were conducted to test the performance accuracy without including special characters, but Macnamara et al. [25] discovered that the performance is not affected

drastically by the exclusion of these characters from the feature set.

As mentioned in Section 2.3, increasing the size of n can increase the accuracy of the classier but beyond a certain level the accuracy decreases. In preliminary tests with a NB classifier it was found that n=6 resulted in the highest accuracy. In addition, the burden on computation and memory usage grows exponentially with n; we have therefore restricted our attention to the cases n=3 and n=6.

3.5 CLASSIFIER IMPLEMENTATIONS

3.5.1 NAÏVE BAYESIAN APPROACH

This implementation of the classifier follows the approach discussed in Section 2.4.2. For each language a vector of n-gram probabilities is computed by

$$\mathbf{l}_j = \frac{\mathbf{f}_j}{|\mathbf{f}_j|}, \tag{3.1}$$

where $\mathbf{f}_j$ is a vector of n-gram frequencies calculated from a language document of class j.

The probability of the test string of size α is calculated in the logarithmic domain. In the next equation the log likelihood simplifies calculations by adding logarithmic probabilities and can be expressed as

$$P(L|D) = \sum_{i=1}^{n-\alpha+1} \ln l_j(c_i), \tag{3.2}$$

where $l_j(c_i)$ is the the probability of the n-gram c_i in the language model $\mathbf{l}_j$.

After calculating the probabilities for all languages, the most likely language profile is selected as the language of the test string. For unseen n-grams a penalty value was assigned. We performed tests using various penalty values and chose the best value based on optimum classification accuracy.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

3.5.2 DIFFERENCE-IN-FREQUENCY CLASSIFIER

The same procedure as in the NB classifier (Section 3.5.1) was followed to create each language profile $\mathbf{l}_j$. For a test string, a vector $\mathbf{x}$ is created in the same manner. Language scores are computed with the equation

$$D_l = \sum_{i=1}^{n-\alpha+1} |l_j(c_i) - x(c_i)|, \tag{3.3}$$

where $l_j(c_i)$ is the the probability of the *n*-gram c_i in the language model $\mathbf{l}_j$ and $x(c_i)$ is the the probability of the *n*-gram c_i in the test vector $\mathbf{x}$.

For each language the above metric is computed and this gives an indication of how similar the test string is to the language model. The language profile with the smallest difference is chosen as the most likely language for the string.

3.5.3 SUPPORT VECTOR MACHINE

The LIBSVM [33] library provides a full implementation of several SVMs. As discussed previously, size of the feature space grows exponentially with *n*, which leads to long training times and extensive resource usage as *n* becomes large; we therefore limited our SVM experiments to *n*=3. A language model was built with samples of size α from a training set. These samples contained a frequency count of each *n*-gram combination in the character string. Thus the feature dimension of the SVM is equal to the number of *n*-gram combinations. Samples of the testing set are created using the same character window as used to build the language model. After training the SVM language model the test samples can be classified according to language.

The SVM used a RBF kernel, and overlap penalties [15] were employed to allow for non-separable data in the projected high-dimensional feature space. Sensible values for the two free parameters (kernel width ($h = 1$) and margin-overlap trade-off ($C = 180$, a large penalty for outliers)) were found on a small set of data. These reasonable parameters were employed throughout our experiments. Classification is done in a one-against-one approach in which $\frac{k(k-1)}{2}$ classifiers are constructed (where k is the number of classes,

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

thus 55 classifiers were constructed) and each one trains from data of two different classes. Classification is done by a voting strategy. Each binary classification is considered to be a vote for the winning class. All the votes are tallied, and the test sample is assigned to the class with the largest number of votes.

3.6 CONCLUSION

In this chapter we discussed our approach and experimental conditions involved. The experimental conditions varied so that the classifiers could be tested under a scope of conditions. We also gave detail about the process for extracting feature data from our collected text data and the classifiers compared in our tests.

CHAPTER FOUR

RESULTS

4.1 INTRODUCTION

In this chapter we discuss our experiments and present the results. The layout is as follows:

- Section 4.2 compares the different the classifiers
- Section 4.3 analyzes the confusion within language families.
- Section 4.4 investigates the assignment of penalty values in the event of unseen n-grams in the training set.

4.2 COMPARING CLASSIFIERS

4.2.1 CROSS VALIDATION

Error rates are calculated using 10-fold cross validation. That is, text data are divided into 10 subsets and for each of the ten folds, a single subset is retained for validation, while a model is trained with the nine remaining subsets. Classification is then performed on the testing data to compute the error rate. The accuracy and error rate are defined as:

$$accuracy = \frac{T_{correct}}{T_{all}} \tag{4.1}$$

and

$$error\ rate = \frac{T_{incorrect}}{T_{all}}, \tag{4.2}$$

where T_{all} is the number of samples classified, $T_{correct}$ is the number of samples correctly classified and $T_{icorrect}$ is the number of samples incorrectly classified. Alternative error measures can also be applied - for example, in [27] the F1-score was used to calculate classification accuracies. We did not consider using this measure, because our test data was evenly distributed among all the classes. If this was not the case, classes with more samples would have influenced the accuracies more. Under these circumstances the F1-score would have given a better indication of the accuracy by measuring a weighted average between the recall and precision rates, as in [27].

The average results from the ten folds are determined to give an overall error estimate. To avoid a bias from having similar text in both the training and test sets, subsets are constructed with minimal shuffling of the original source documents. Therefore, different subsets will tend to contain text from different domains.

4.2.2 EVALUATION FACTORS

Classification performance depends on the factors discussed in Section 1.2. Our study investigates these factors, by comparing classifiers under different conditions. Cross validation was done using the same text for all three classifiers. Tests were performed on training text sizes of 200K, 400K, 800K, 1600K and 2000K. In a comparison test, the size of the character window was the same for each classier; these windows were 15, 100 and 300 characters long. The only factor that could not be explored fully was the size of the *n*-gram, *n*=3 and *n*=6. It was possible to build likelihood and difference-in-frequency classifiers with *3* and *6*-grams, but due to slow processing and excessive memory requirements for high feature dimensions the SVM is built with *3*-grams only.

4.2.3 EXPERIMENT

For the 15 character window the *6*-gram NB model performs best at all but one dataset. The SVM trained with *3*-grams did slightly better than the *6*-gram NB model with the same

dataset of 200K characters. However, for this window size, the differences between the SVM and the *6*-gram NB classifier are quite small. The SVM does noticeably better than the *3*-gram NB model for the same datasets (for a 2M character dataset the difference is 5.13%). The *6*-gram difference-in-frequency model performs worse than the *3*-gram NB model and the *3*-gram difference-in-frequency model is significantly worse everywhere. The accuracies of the competitive models are still improving when 2M training data are employed (the largest amount we could employ).

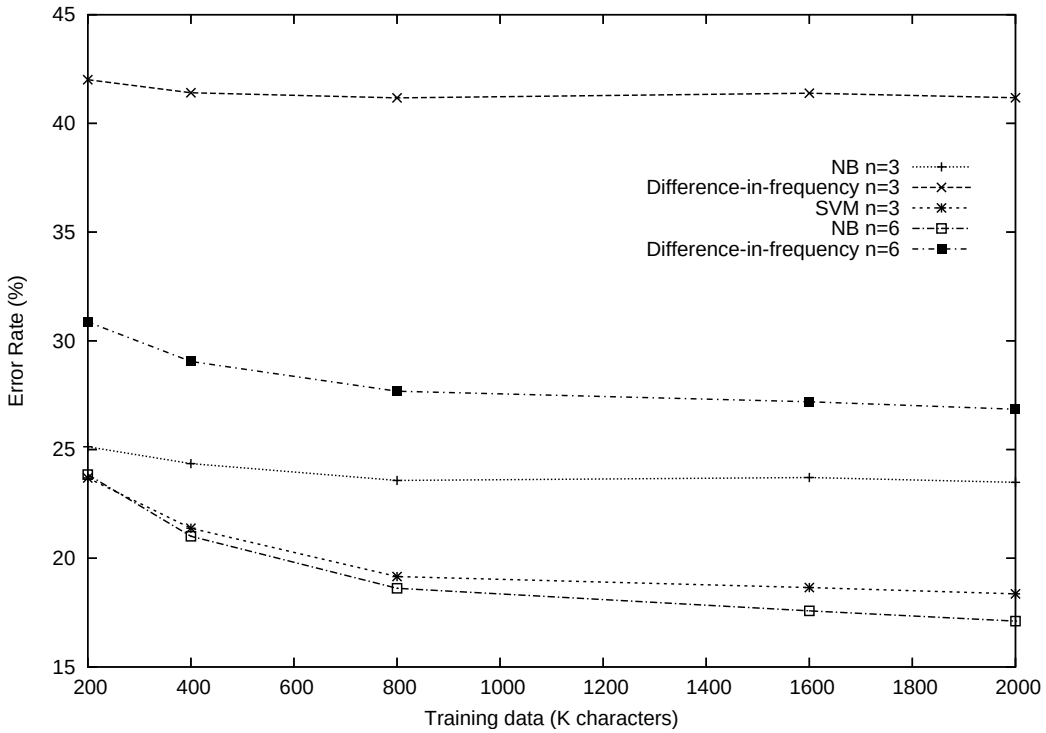

Figure 4.1: Classification results on 15 character window. Error rates are calculated by applying 10-fold cross-validation on various amounts of text.

	NB (n=3)	DF (n=3)	SVM (n=3)	NB (n=6)	DF (n=6)
200k	25.13	42.01	23.69	23.85	30.87
400k	24.35	41.11	21.38	21.01	29.05
800k	23.58	41.17	19.16	18.62	27.68
1.6M	23.71	41.39	18.65	17.58	27.2
2M	23.49	41.18	18.36	17.11	26.85

Table 4.1: Classifier error rates for various amounts of training text (in number of characters). Window size = 15 characters. Difference-in-frequency is abbreviated as DF.

For a 100 character window, the outperformance of the *6*-gram NB model compared to the SVM is more significant than with a 15 character window test. The average percentage difference is 1.82%, where it was 0.68% in the previous test. Now, the SVM and *3*-gram NB model have very similar performances from 200K to 800K characters and from 1.6M characters the difference is more evident but less than with the 15 character window (now only 1.06% at the 2M character set). It was interesting to find that the NB classifier performed best for only the smallest dataset. The performance of the difference-in-frequency classifier cannot be compared to the NB classifier and SVM. For the case of n=3 there is not much of a difference in accuracy performance for the different datasets. Again it seems that the accuracies continue to improve uniformly with the amount of training data available, for the competitive classifiers.

	NB (n=3)	DF (n=3)	SVM (n=3)	NB (n=6)	DF (n=6)
200k	5.91	12.97	6.26	3.91	11.63
400k	4.72	12.14	4.65	2.46	9.77
800k	4.07	11.95	3.54	1.81	8.88
1.6M	4.1	12.47	3.14	1.7	8.48
2M	3.92	12.17	2.93	1.53	8.21

Table 4.2: Classifier error rates for various amounts of training text (in number of characters). A window size = 100 characters.

Figure 4.3 contains the results for the largest window size; here, the improvement in accuracy with increasing amounts of training data is less uniform, suggesting that the accuracies of the classifiers may be saturating at these levels. The *6*-gram NB classifier

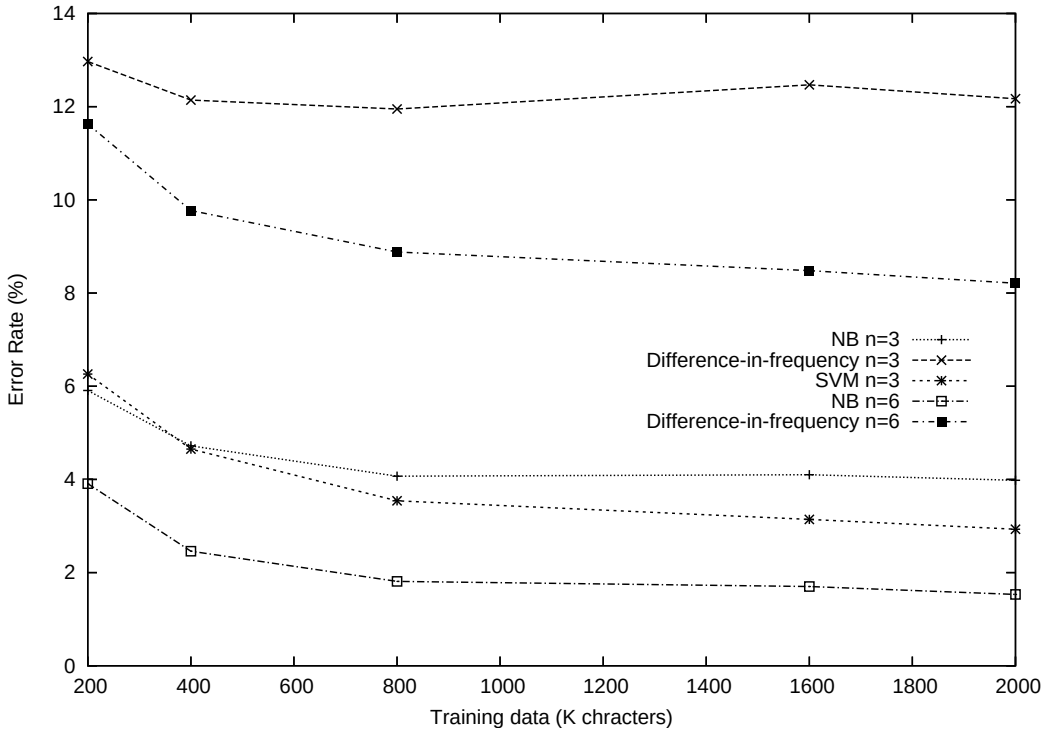

Figure 4.2: Classification results on 100 character window. Error rates are calculated by applying 10-fold cross-validation on various amounts of text.

performs best overall. In this dataset, the *6*-gram NB does better than the SVM by 0.52% on average, and at convergence the SVM performs 0.48% better than the *3*-gram NB model. As with the 100 character window, the *3*-gram NB model slightly outperforms the SVM only at the 200K dataset. The difference-in-frequency classifier was once again outperformed. The *6*-gram difference-in-frequency classifier showed an average 2.77% accuracy deficit when compared to the best NB classifier with 800K and more training data.

	NB (*n*=3)	DF (*n*=3)	SVM (*n*=3)	NB (*n*=6)	DF (*n*=6)
200k	2.57	6.33	2.77	1.44	5.21
400k	1.88	5.63	1.74	0.62	3.82
800k	1.46	5.28	1.18	0.46	3.4
1.6M	1.68	5.66	1.2	0.67	3.39
2M	1.65	5.5	1.1	0.6	3.25

Table 4.3: Classifier error rates for various amounts of training text (in number of characters). A window size = 300 characters.

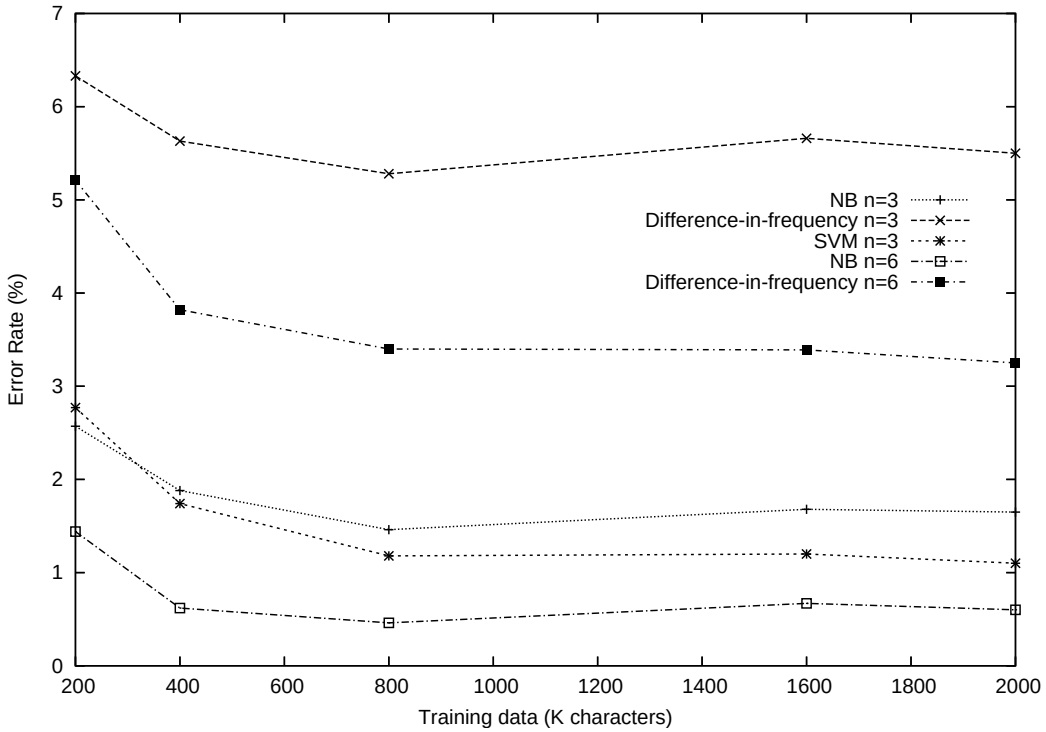

Figure 4.3: Classification results on 300 character window. Error rates are calculated by applying 10-fold cross-validation on various amounts of text.

4.2.4 DISCUSSION

From the results it is obvious that larger test windows increase classification accuracy, to the extent that the *6*-gram NB classifier with a 2M training set and a character window of 300 achieved an error rate of 0.6%. For the smallest test sample accurate classification was difficult, and the lowest error rate of 17.11% was found with the *6*-gram NB model trained with the largest dataset. In Section 4.3 we will analyze our data more and discover the reason for these high error rates. For the intermediate window of 100 characters, the *6*-gram NB again performed best, with a 1.53% error rate for a 2M character training set.

The size of the training set improved the accuracy in most cases. Except for the largest window sizes, the best classifiers continued to improve even when 1.6M training characters per language are available. For small windows, the SVM is apparently more efficient in its use of the training data than the NB classifier (i.e. for the least amount of training data the n=3 SVM and n=6 NB classifier are almost equally accurate, but for larger training sets the NB classifier improves more rapidly).

It is also interesting to track the classifiers performance across different window sizes for a fixed amount of training data. Figures 4.4 and 4.5 represent these results using the smallest and largest training sets. For the small training set and a 15 character window the SVM performs best, but then slightly worse than the *6*-gram NB model at the larger windows. For the larger dataset the 6-gram NB model outperforms all the classifiers for all character windows. The SVM outperforms the *3*-gram NB model at all character windows and training-set sizes. Though the performance of the difference-in-frequency classifier is consistently the worst, it demonstrates the same broad patterns of change with n, and training-set and window sizes. In Section 4.4 we show that a relatively small modification to the difference-in-frequency classifier improves its performance significantly. Although the basic implementation of this classifier is consistently inferior to NB, we suspect that the two classifiers are virtually indistinguishable when this more refined implementation is used.

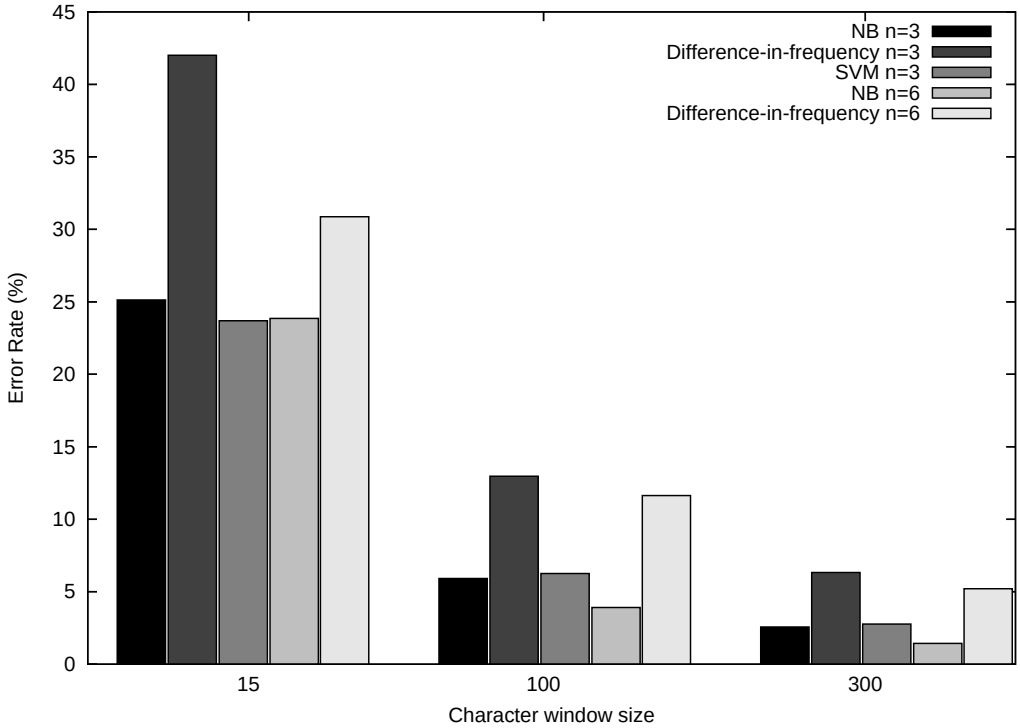

Figure 4.4: Error rates for different characters windows. This results were obtained using 10-fold cross-validation on 200K characters of text.

4.3 CONFUSION WITHIN LANGUAGE FAMILIES

4.3.1 CONFUSION MATRIX

In Section 4.2.3 we found significantly higher error rates on the smallest (15-character) window. To understand the results better we analyzed the output of a 10-fold cross validation on a 200K dataset using the SVM classifier. Using a confusion matrix (Table 4.5) we could investigate the types of errors that occur. Each row represents the correct language of a set of samples. The columns indicate the languages selected by the classifier. Thus, more samples on the diagonal of the matrix indicate better overall accuracy of the classifier. The abbreviations used in the confusion matrices can be found in Table 4.4. It is clear that the vast majority of errors result from confusions within the Sotho and Nguni language families.

We then created a confusion matrix based on the confusable language families in Table 4.5. (Since Germanic languages were reasonably discriminated, we did not group them together.) The overall performance increases drastically and this indicates that language families can successfully be classified with only 15 input characters.

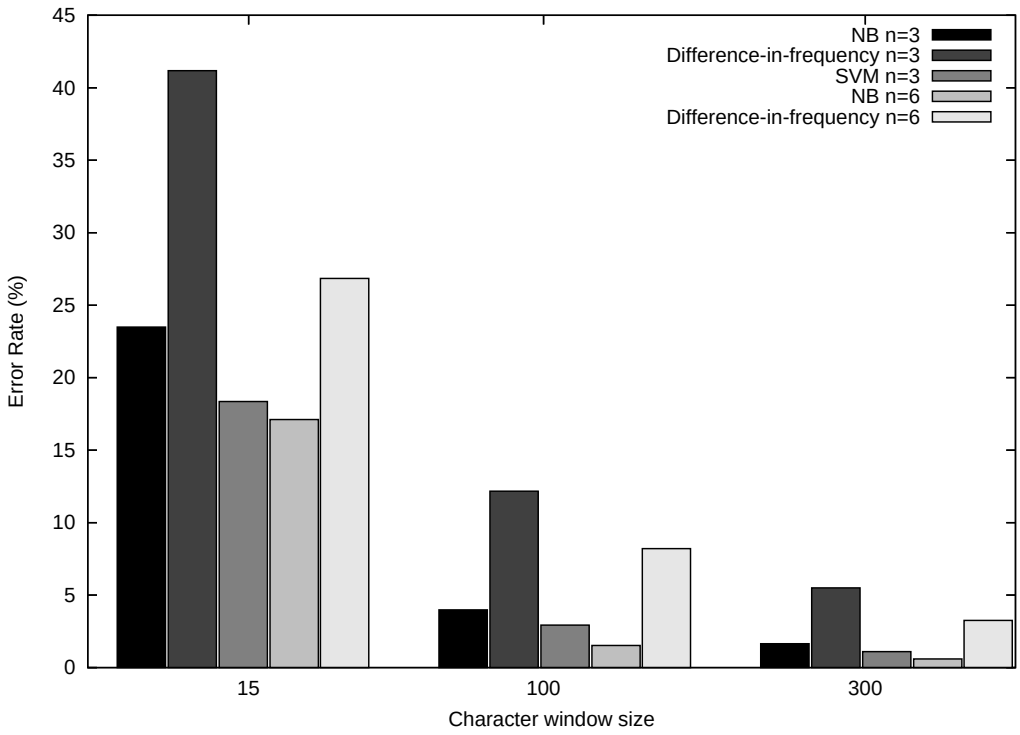

Figure 4.5: Error rates for different characters windows. This results were obtained using 10-fold cross-validation on 2M characters of text.

Language or language family	**Abbreviation**
Sesotho	Ses
Sesotho Sa Leboa	Sep
Setswana	Set
isiXhosa	Xho
isiZulu	Zul
isiNdebele	Nde
Siswati	Swa
Tshivenda	Ven
Xitsonga	Tso
Afrikaans	Afr
English	Eng
SF	Sotho Family
NF	Nguni Family

Table 4.4: Language and language family abbreviations

Ses	Sep	Set	Xho	Zul	Nde	Swa	Ven	Tso	Afr	Eng	
9743	1370	1589	36	50	41	32	75	75	28	68	Ses
1698	**9237**	1906	34	50	41	14	49	75	15	53	Sep
1991	1994	**8843**	25	23	45	32	36	58	29	36	Set
72	32	15	**8123**	2441	1821	434	44	69	41	50	Xho
52	42	16	2769	**7177**	2192	663	54	69	30	83	Zul
82	59	42	2343	2692	**7157**	594	98	115	12	47	Nde
70	26	33	600	851	647	**10622**	41	122	24	137	Swa
142	80	67	139	90	158	53	**12158**	270	15	46	Ven
138	124	77	124	87	106	161	250	**12028**	22	78	Tso
27	14	17	30	25	12	25	9	11	**12876**	232	Afr
44	38	9	34	53	27	51	21	45	177	**12608**	Eng

Table 4.5: Confusion matrix for the SVM classifier trained with 200K characters, for n=3 and a window size of 15 characters.

Ses	Sep	Set	Xho	Zul	Nde	Swa	Ven	Tso	Afr	Eng	Overall
25.67	29.87	32.56	38.05	45.41	45.5	19.37	8.02	8.84	3.03	3.81	23.69

Table 4.6: Error rates for languages calculated from confusion matrices in Table 4.5

SF	NF	Ven	Tso	Afr	Eng	
38371	423	160	208	72	157	SF
541	51096	237	375	107	317	NF
289	440	12158	270	15	46	Ven
339	478	250	12028	22	78	Tso
58	92	9	11	12876	232	Afr
91	165	21	45	177	12608	Eng

Table 4.7: Confusion matrix for the SVM classifier trained with 200K characters, for n=3 and a window size of 15 characters. Nguni and Sotho languages are combined into families.

SF	NF	Ven	Tso	Afr	Eng	Overall
2.59	2.99	8.02	8.44	3.03	3.81	4.88

Table 4.8: Error rates for cross language families calculated from confusion matrices in Table 4.7.

4.3.2 A GRAPHICAL REPRESENTATION OF LANGUAGE DISTANCES

In the previous experiment, the confusion matrices provided a clear indication of the ways the languages group into families. These relationships can be represented visually using graphical techniques. Multidimensional scaling (MDS) is a technique used in data visualization for exploring similarities or dissimilarities in data. The algorithm uses a matrix of similarities between items and then assigns each item a location in a low dimensional space to match those distances as closely as possible. We used the confusion matrix to serve as similarity measure between languages, using the statistical package XLSTAT [34]. The confusion matrix was processed into a matrix of distances using the Pearson correlation coefficients between the rows, and input into the multidimensional scaling algorithm which mapped the language similarities in a 2-dimensional space.

Figures 4.6 and 4.7 show the mapping that was created using confusion matrices resulting from two different experimental conditions; in both cases we can see that the languages from the same subfamilies cluster together. The Nguni and Sotho languages are more closely related internally than the pair of Germanic languages and within the Nguni language Siswati is somewhat distant from the other three languages. As expected,

Tshivenda and Xitsonga are consistently separated from the other nine languages.

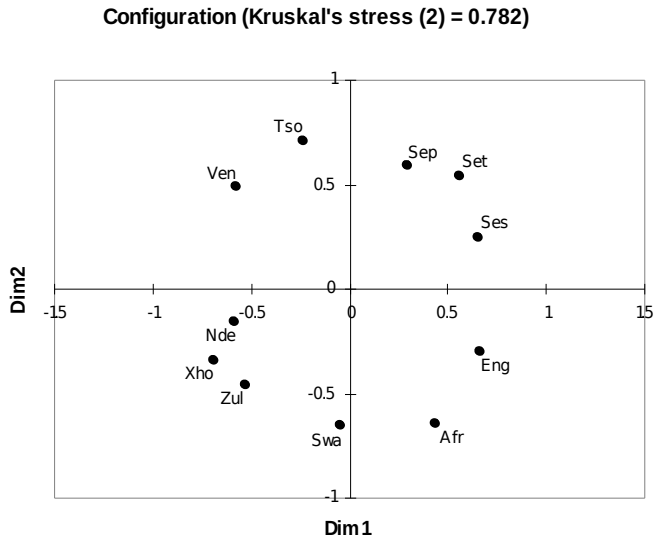

Figure 4.6: Multi-dimensional scale representing similarities between languages. The similarities were calculated from the confusion matrix of the SVM classifying a 200K dataset on a 15 character window.

Appendix B contains several examples of the graphical representations that result when other confusion matrices are employed.

4.4 PENALTIES FOR UNSEEN *N*-GRAMS

In the NB framework, *n*-grams that are not present in the training set would by default receive zero probability, and therefore assign zero posterior probability to a language regardless of other information. This is an artifact of the finite size of practical training sets, and a more appropriate method for the treatment of unseen *n*-grams is to apply a large but finite penalty. The penalty value specifies the frequency that is assigned to an unseen *n*-gram. Initial experiments showed that such a penalty can improve classification results. In previous studies unseen *n*-grams were penalized in different ways. Examples are:

- Cavnar and Trekle [5] added a maximum out-of-place value to the distance measure calculation.
- In [14] a value of a half is added to events seen in at least one training set, but not seen in a particular language.

- For a NB approach, Padro and Padro [12] assigned a fix number of occurrences to the *n*-grams not in the training set.

- For these events, Binas [23] applied Laplace smoothing when the simple Markov model was used.

  None of the above approaches are convincingly motivated on theoretical or practical grounds. We therefore investigated how the choice of penalty value affects the accuracy of the NB classifier. Experiments were conducted on both the *3*-gram and *6*-gram classifiers. All three character windows were evaluated. Tests were performed on the 200K training set, and with later experiments the best penalties found on this dataset also proved best on all the other training sets.

4.4.1 *3*-GRAM NB CLASSIFIER

Figures 4.8-4.10 show the results of the experiments performed with different character windows. (Note that different scales are used in these figures.) For each character window the error rates at various penalty values are shown, along with the penalty value which gives the lowest error rate. It is clear that the larger the character window, the smaller the optimum penalty score; therefore, unseen *n*-grams are penalized more for larger windows.

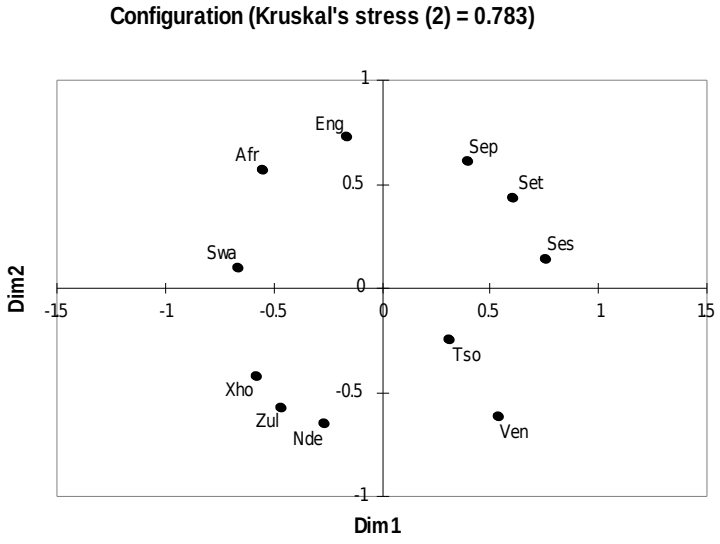

Figure 4.7: Multi-dimensional scale representing similarities between languages. The similarities were calculated from the confusion matrix of the SVM classifying a 2M dataset on a 15 character window.

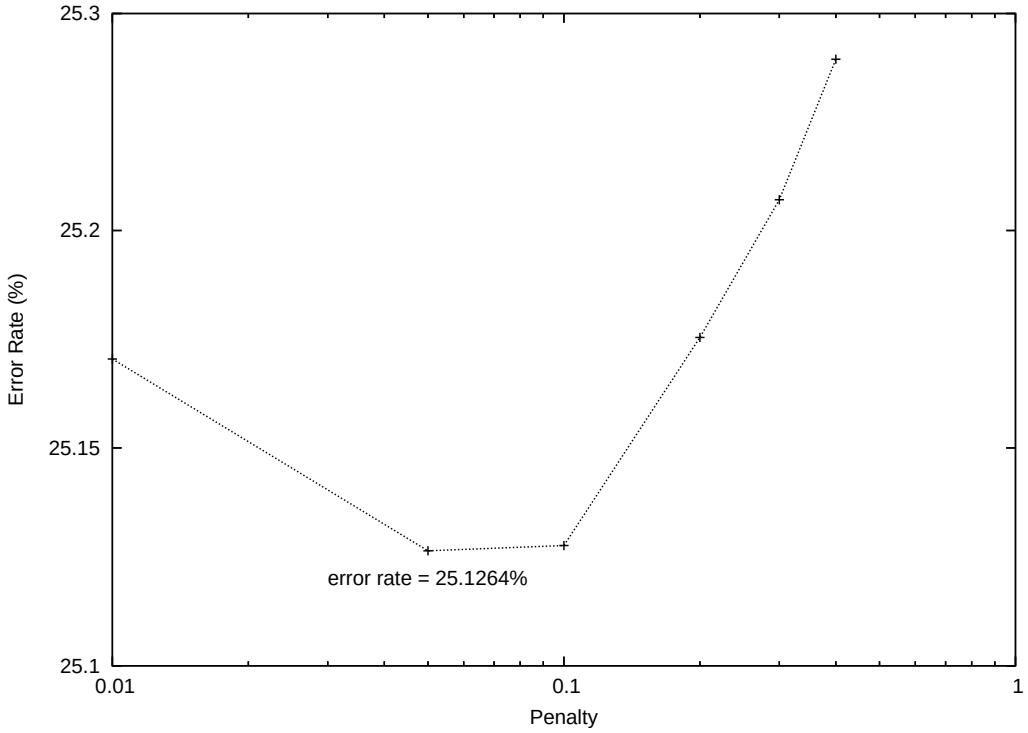

Figure 4.8: Error rates vs. penalty scores for the n=3 NB classifier. The test were performed on a 200K character set and a window size of 15 characters.

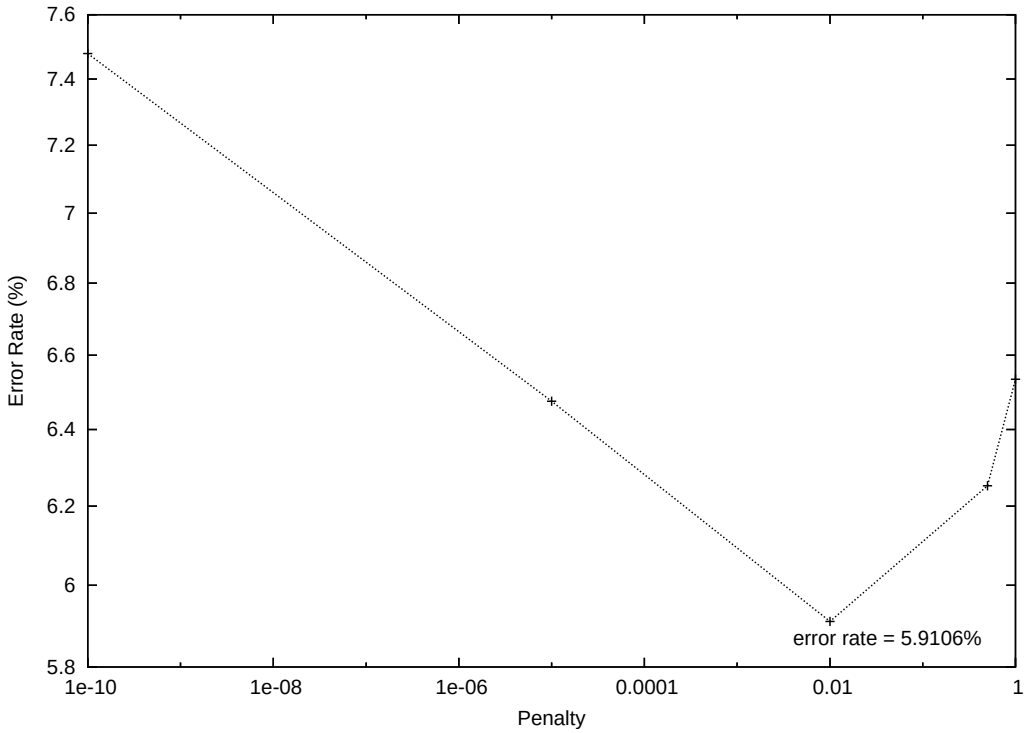

Figure 4.9: Error rates vs. penalty scores for the n=3 NB classifier. The test were performed on a 200K character set and a window size of 100 characters.

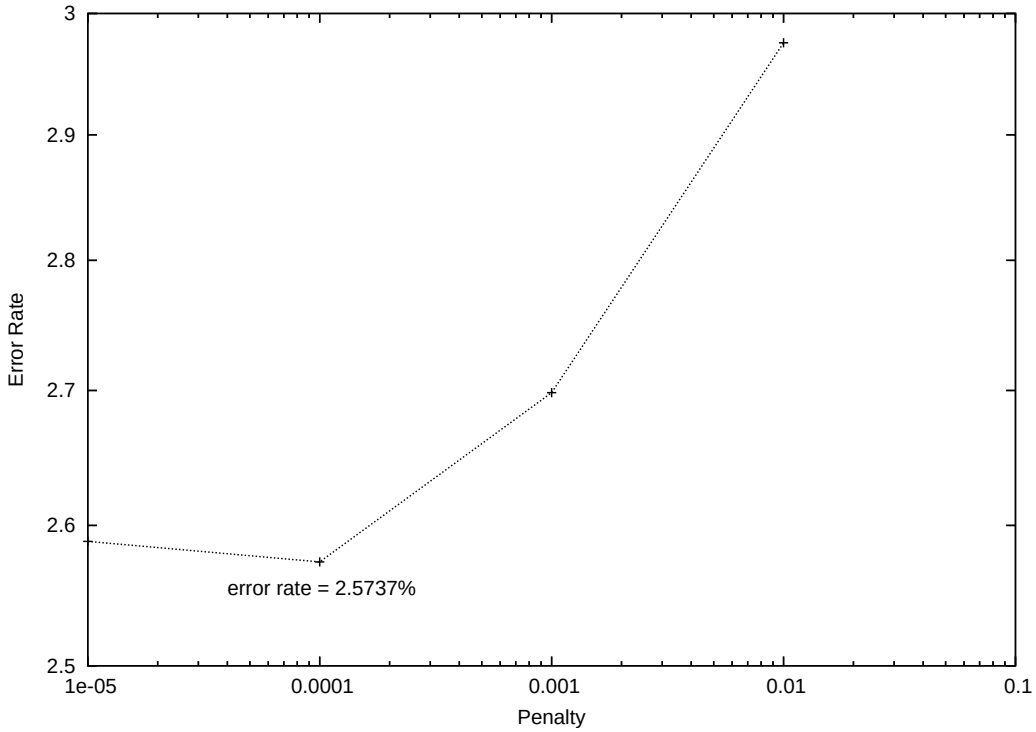

Figure 4.10: Error rates vs. penalty scores for the n=3 NB classifier. The test were performed on a 200K character set and a window size of 300 characters.

4.4.2 *6*-GRAM NB CLASSIFIER

The results of the same experiments with n=6 are shown in Figures 4.11-4.13. As with the *3*-gram model there is an optimum penalty value for each of the character windows. These values also decrease as the size of the character window increases.

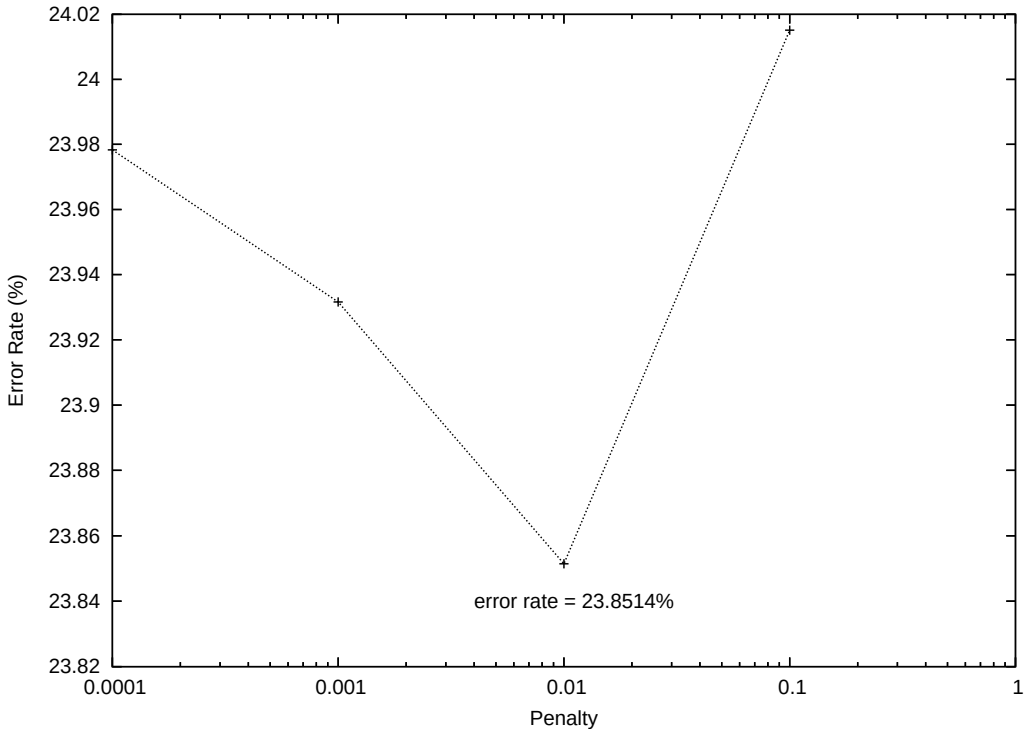

Figure 4.11: Error rates vs. penalty scores for the n=6 NB classifier. The test were performed on a 200K character set and a window size of 15 characters.

4.4.3 THE EFFECT OF *N*-GRAM SIZE ON OPTIMAL PENALTY SCORE

Figure 4.14 shows the optimum penalty values for different character windows for both n=3 and n=6. In both cases, a lower penalty is assigned for larger character windows. This means that unseen n-grams are penalized more for larger character windows. Compared to the *3*-gram model the *6*-gram model is penalized significantly more heavily when unseen n-grams occur. Both of these trends are somewhat surprising, and are currently being investigated theoretically.

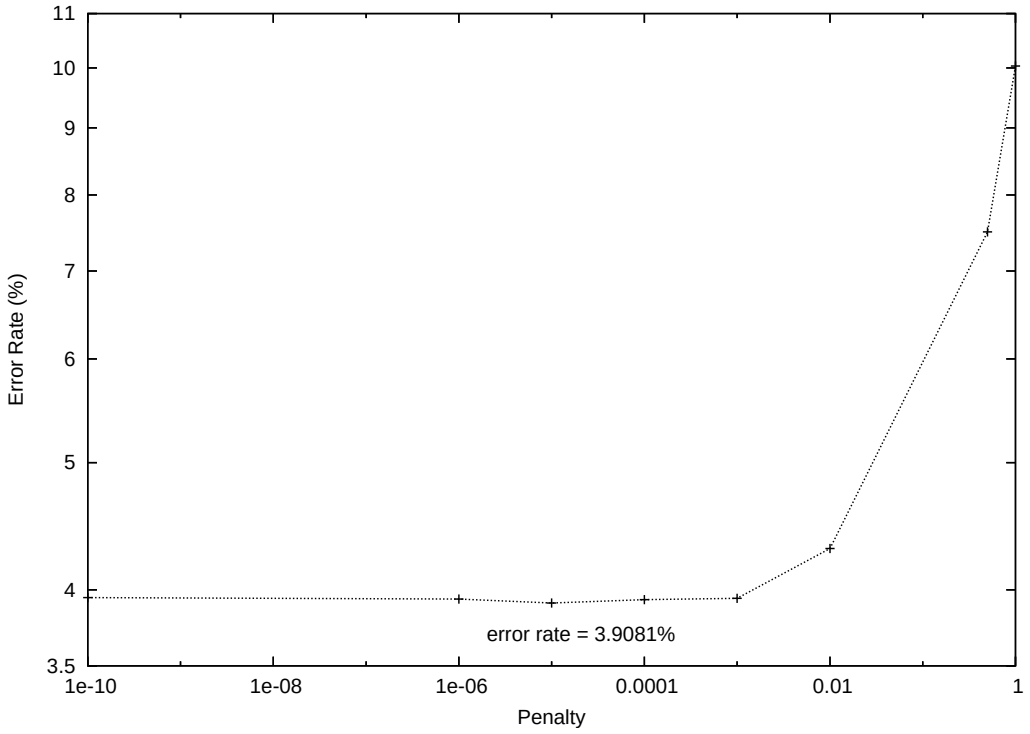

Figure 4.12: Error rates vs. penalty scores for the n=6 NB classifier. The test were performed on a 200K character set and a window size of 100 characters.

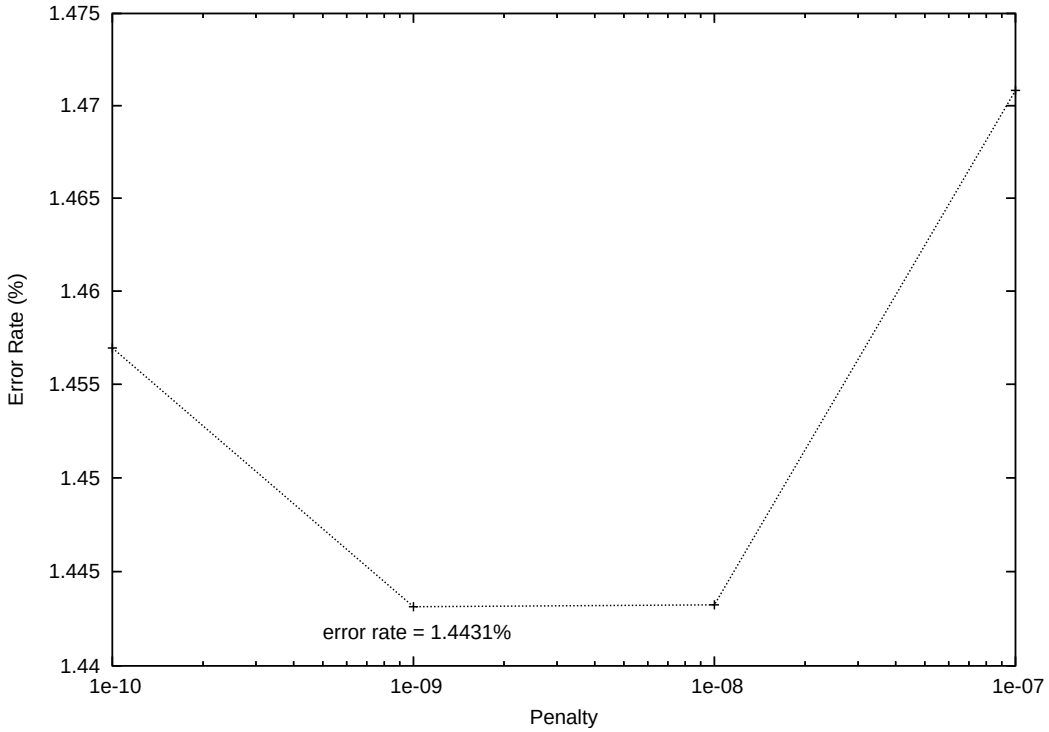

Figure 4.13: Error rates vs. penalty scores for the n=6 NB classifier. The test were performed on a 200K character set and a window size of 300 characters.

4.4.4 DIFFERENCE-IN-FREQUENCY CLASSIFIER

In Figure 4.15 we show the accuracy of the difference-in-frequency classifier using various penalty values. A *6*-gram difference-in-frequency classifier was tested using the 200K training set and a 100 character window. As with the previous experiments, the penalty value specifies the frequency that is assigned to an unseen n-gram. The graph shows that the optimal performance is achieved for a wide range of values.

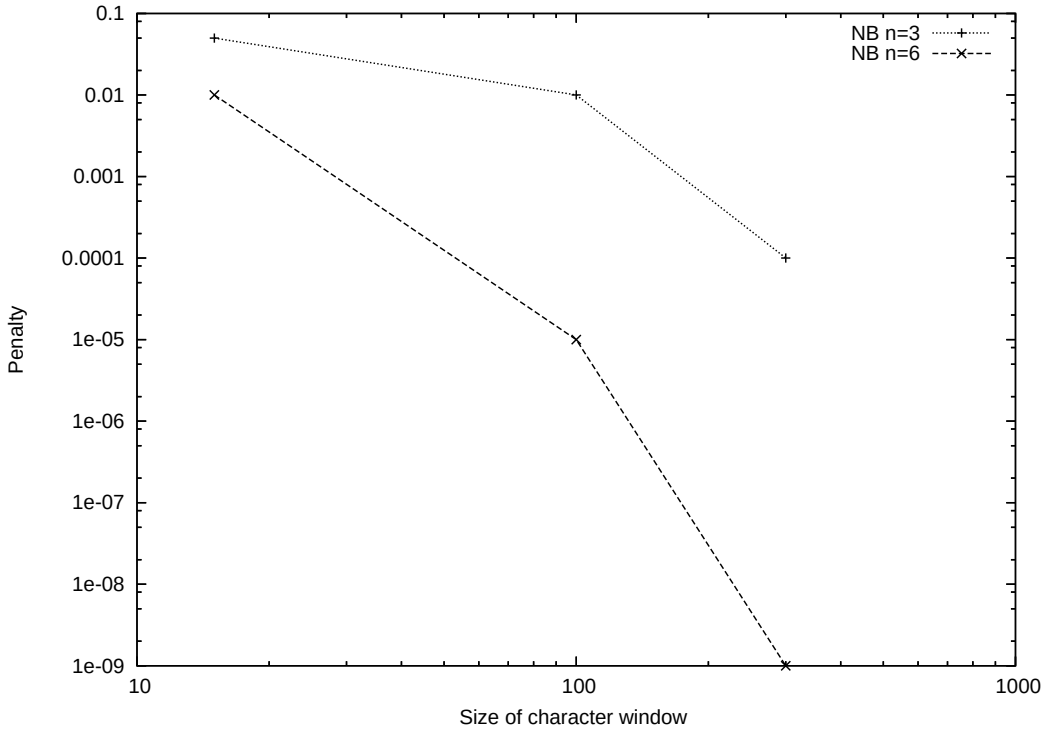

Figure 4.14: The optimum pentalty score vs. error rates for the NB classifiers.

A different method for assigning the penalty score was also investigated. In the event of an unseen *n*-gram we assigned a fixed penalty value, and this value is added when we calculate the metric with Equation 3.3. In Figure 4.16 and 4.17 tests were performed on a 100 and 300 character window with the *6*-gram difference-in-frequency classifier. The results show a large improvement. On a 100 character window the lowest error rate was 4.05%, where 11.63% was found for a frequency penalty of 0 (Table 4.2). For a 300 character window the best penalty gave an error rate of 1.4%. This is also much better than the results in Table 4.3 (5% error rate at a frequency penalty of 0). Thus, the difference-of-frequencies classifier functions much more accurately if the unseen *n*-grams are penalized with a fixed value rather than a difference value, even outperforming NB by a small margin for *n*=6.

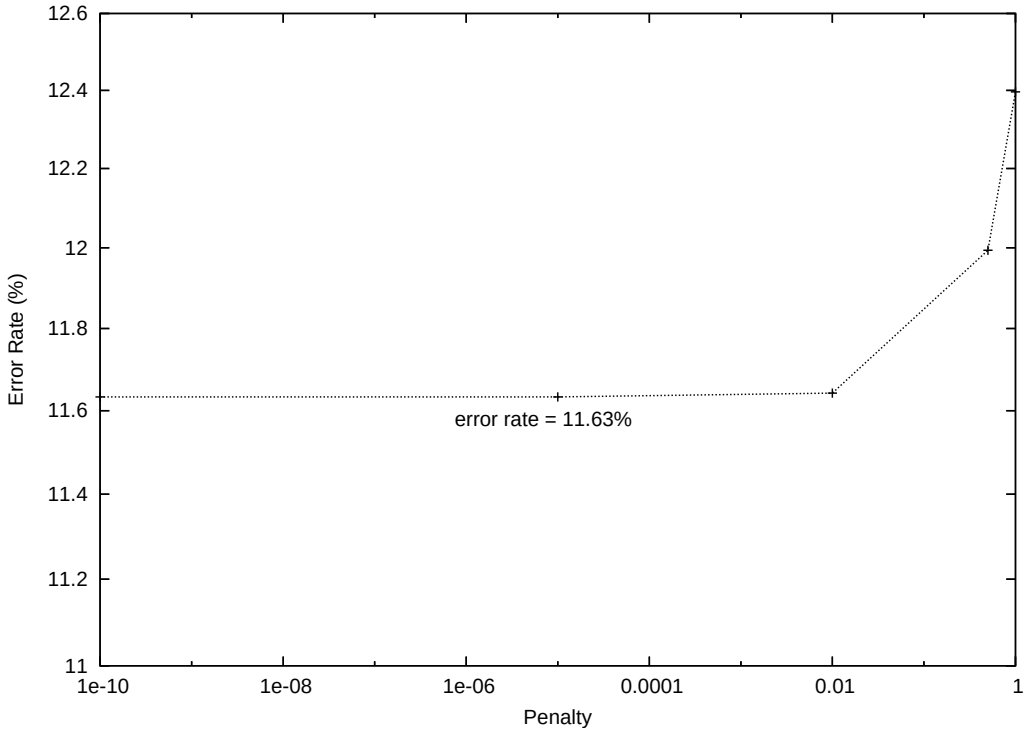

Figure 4.15: Error rates vs. penalty (relative) scores for the n=6 Difference-in-frequency classifier. The test were performed on a 200K character set and a window size of 100 characters.

4.5 COMPARISONS WITH OTHER STUDIES

This section compares our results with those found in previous studies. As mentioned, earlier studies were conducted and evaluated under different circumstances. It is nevertheless possible to make some sensible conclusions.

In the *3*-gram rank ordering study [5], 98.6% accuracy was found for a test string smaller than 300 characters. The training sets varied between 20K and 120K characters. The closest comparison we can make is if we compare the results with our smallest dataset and consider the performance accuracies between 100 and 300 characters. For 100 characters our *3*-gram NB obtains an accuracy of 94.1% and the same method using *6*-grams 96.1%. For the 300 character window the *3*-gram NB accuracy is 97.4% and the accuracy of the *6*-gram method is 98.6%. The accuracies are in similar ranges. Previous studies showed that the NB approach outperformed the *n*-gram rank ordering method when evaluated under the same conditions [12].This suggests that the set of languages used in [5] were easier to

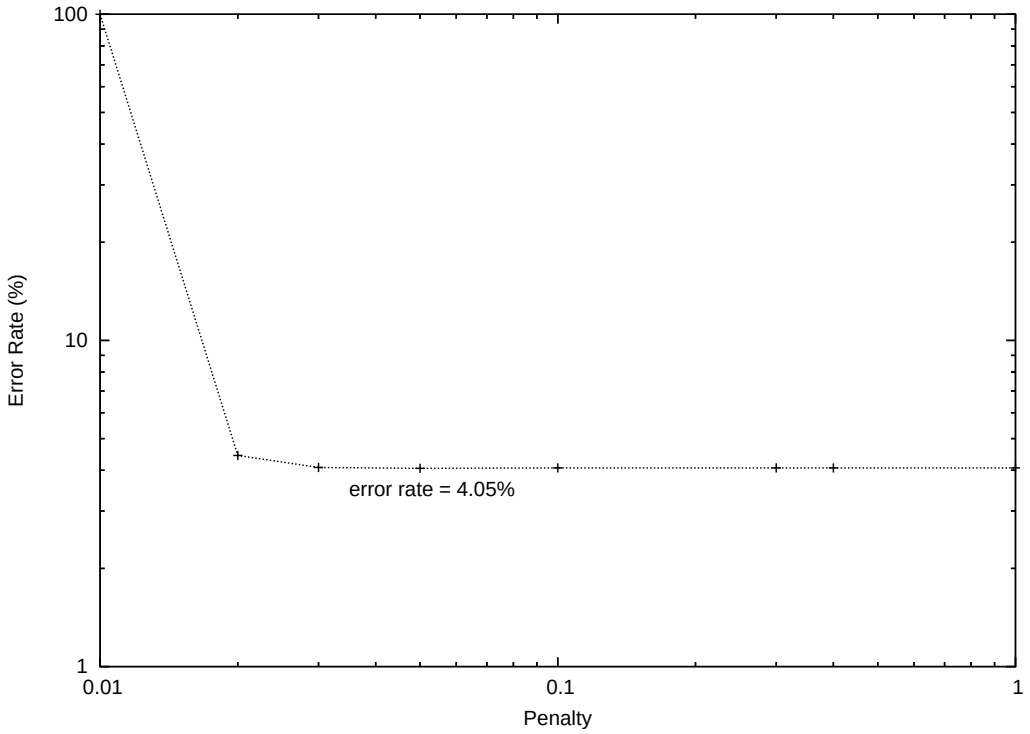

Figure 4.16: Error rates vs. penalty (fixed) scores for the n=6 difference-in-frequency classifier. The test were performed on a 200K character set and a window size of 100 characters.

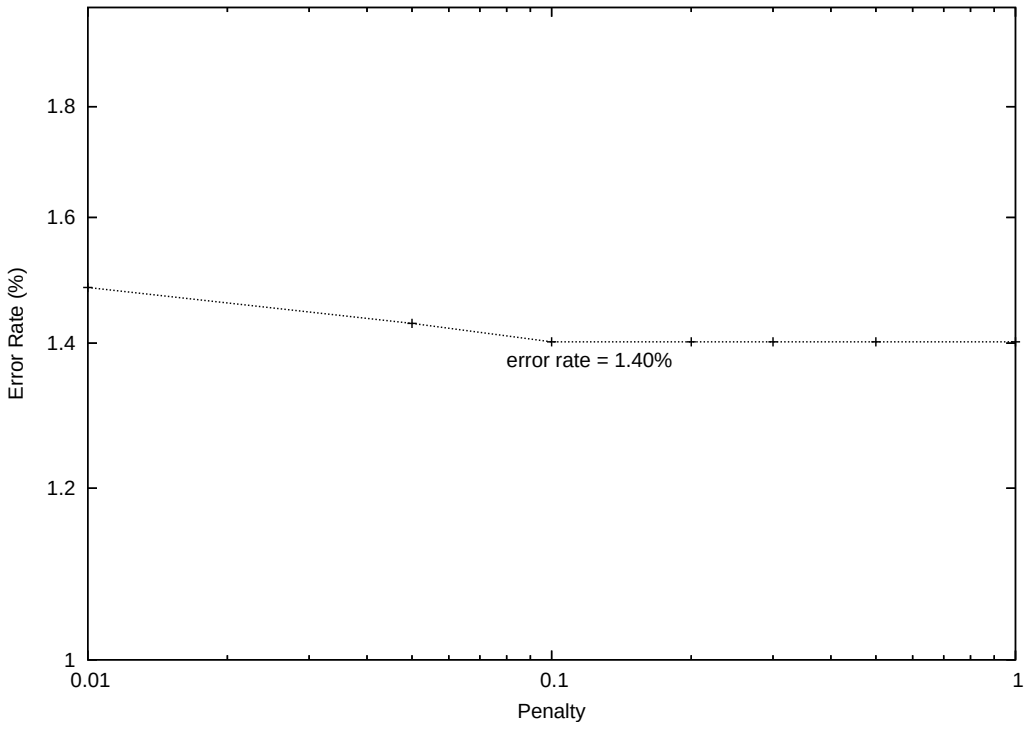

Figure 4.17: Error rates vs. penalty (fixed)scores for the n=6 difference-in-frequency classifier. The test were performed on a 200K character set and a window size of 300 characters.

classify than the languages used in our study.

In [6] the NB approach was applied to text-based LID. Two languages were addressed and both were from different family groups. For a 50K training set and a 20 character test window 92% accuracy was found. If we compare this with our cross language experiment in Section 4.2.3 and a 15 character window, 95.1% accuracy was achieved with the SVM classifier. Though our training set was larger, more languages were involved in the classification. The study also found that larger training data and character windows improve classification.

The NB approach was also evaluated in [9]. Sixteen languages were involved in the experiments but it was not clear what the amount of text was for building the model. For a 1-2 word test window an overall accuracy of 76.5% was achieved. Our *3*-gram NB classifier has an accuracy of 74.9% when trained with our smallest dataset. Under the same conditions the *6*-gram NB improves the accuracy to 76.4%. These values seem comparable if we assume similar conditions.

In [11] the dot product approach was followed. Thirteen languages with average of 70KB per language were used to train the classifier. Word and *4*-gram statistics were combined and used as features for building the models. The average performance was 85.4% for a 20 character input and over 99% for 130 characters and up. If we compare our *6*-gram NB classifier (considering that [11] trains with an improved NB classifier) with the largest dataset, the results in the two studies are similar. For a 15 character window the classification accuracy is 82.9% and for testing on a 100 character window a 98.5% accuracy is reported.

In [12] text-based LID was tested on 6 languages. It was found that training sizes larger than 50K words did not improve the accuracy of the classifiers. If we consider the Sotho languages (average 5 characters per word) this would have occurred at approximately 250K characters and for the Nguni languages (average 8 characters per word) at 400K characters. Our experiments showed that the accuracy rates had not converged at a much

larger dataset of 1.6M characters.

Eighteen languages were classified in [14] using the relative entropy approach and bi-gram statistics. They use the number of lines to measure the data size. We approximate 1 line as a 100 character window. For a 200K training set accuracy of 94.1% was achieved on a 100 character window. Our 3-gram NB classifier performs equally under the same conditions.

In [28] three classifiers were compared and were tested on five languages. Models were trained on an average of 25KB of text per language. On a testing set of 100 character samples, an RBF classifier showed 99.2% accuracy, a neural network performed with 99.7% accuracy and a neural network implemented with a different kernel performed with 99.2% accuracy. For a 200K dataset our 6-gram NB classifier performs 3.6% less accurately than the best of these classifiers, but much closer with the largest dataset. For 20 character samples the RBF accuracy dropped to 71% accuracy, the neural network 47.2% and the neural network with their kernel method dropped to 88.4% accuracy. For a character window of 15 our best classifier performed better than two less accurate classifiers. For a 2M dataset our best classifier performed 5.5% less accurately than the better neural network. Again, the different languages and datasets employed mean that these numbers are not directly comparable.

In [13] performance was tested based on 5-fold cross validation on 17 languages and an average of 578 sentences per language. On a test sample with an average of 50 characters per test string, *n*-gram rank ordering performed with 90.2% accuracy, the centroid method with 95.9% accuracy and the SVM gave an overall best performance accuracy of 99.7%. If we consider a sentence as a 100 character window we approximate their dataset as 500KB per languages. For a 400K dataset our SVM shows a lower performance accuracy of 95.35% and for a 2M dataset 97.7% (we assumed a 100 character window).

In [23] three type of Markov models were tested on 6 languages. Models were trained with 5K of text. The AGMM proved best on a string of 10 characters, with an accuracy of

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

74.13%. For a 15 character window our *3*-gram NB model shows an accuracy of 74.87%.

4.6 CONCLUSION

In this chapter we have discussed our experiments and results. The detailed procedure of our evaluation process was given. This included our comparison measure and experimental conditions. We analyzed the effect of the size of the training data, the test string and of n on performance accuracies.

All the classifiers showed improvement at larger training sets. From the smallest dataset to the 1.6M dataset, a considerable reduction in error rate occurred in most cases. Error rates between the 1.6M and 2M datasets are more similar, but a statistically significant improvement is again present (except for the 300-character window).

Overall, we found that the *6*-gram NB classifier performed best except for one experimental condition: the *3*-gram SVM did slightly better than this classifier with a 200K dataset and a 15 character window. Also, for the other training sets and the 15 character window, the *6*-gram NB did not do significantly better than the SVM.

The SVM performed better than the other two *3*-gram classifiers except under two circumstances. The *3*-gram NB classifier did somewhat better with the smallest dataset using a 100 and 300 character window.

The difference-in-frequency classifier did not prove a challenge for the other two methods. The classifier also showed that the *6*-gram statistics improves accuracy over the smaller *n*-gram size.

Another significant advantage of the NB classifier is that new language documents can simply be merged into an existing classifier by adding the *n*-gram statistics of these documents to the current language model. For the SVM a whole new classifier would need to be trained.

The poor performance on 15 character window was analyzed using a confusion matrix. The results indicated clearly that confusion is found within language families (Nguni and Sotho), which results in high error rates. A confusion matrix was then created where all the languages in a family were merged, which drastically improved the classification accuracy. For the larger character windows the confusion within language families became significantly less. Thus for the smallest character window it is possible to identify the language families by examining where there is a notable uncertainty between languages in the confusion matrix. For the larger character windows the classifiers found it much easier to discriminate between languages within a family. These relationships between languages were represented graphically using MDS.

It is possible that an n-gram occurs in a test string but not in a training set. To assume a 0 probability for the n-gram is too extreme; therefore an appropriate penalty score should be assigned in this case. We experimented with various penalty values for the NB classifier, and found that for a larger character window an unseen n-gram needs to be penalized more. The size of n also plays a role and the 6-gram models needs to be penalized more. The difference-in-frequency classifier assumes 0 probability if an n-gram does not occur in the training set. The accuracies of this classifier can be improved considerably if a fixed penalty value is added for unseen n-grams.

CHAPTER FIVE

CONCLUSION

5.1 INTRODUCTION

This chapter summarizes the conducted research. For completion, final contributions and future work is discussed. The layout is as follow:

- Section 5.2 summarizes our research.
- Section 5.3 discusses some of the contributions of this work.
- Section 5.4 provides a number of proposals for future work.

5.2 SUMMARY OF RESEARCH

In this research, factors affecting accuracy in text-based LID using *n*-gram statistics were investigated. The factors that were addressed include:

- The type of classifier employed
- The amount of training data
- The size of the string to be identified

- The feature dimension space, which is dependent on the size of n
- The similarity of the languages to be identified

The 11 official languages of South Africa were used for the study. These languages belong to a number of closely-related families, which makes classification within family groups difficult.

Three classifiers were compared in this study:

- The NB (otherwise known as the simple Markov model or likelihood based classifier) which achieved good results in previous studies.
- A difference-in-frequency classifier, which is based on a simple difference measure.
- The SVM, which is a more complex classifier.

N-gram statistics were used as features for classification. The accuracy of classification improves for a larger n but an optimum n does exist. Initial experiments showed that n=6 gives best performance. The NB and difference-in-frequency classifier were trained with both *3*-grams and *6*-grams. We were only able to train the SVM with *3*-grams because of the large feature space using 6-grams which result in a complex classifier that takes an unrealistic amount of time to train.

Tests were performed on three different sizes of character windows. These sizes were chosen to provide a range of challenges for classification (the larger the test string, the easier the classification task becomes).

Text from various domains was collected and was preprocessed before building our models. The data was divided into sets of different sizes and the classifiers were trained with the different training sets. All our classifiers were tested under the same conditions. We evaluated the performance by applying 10-fold cross validation. The results were as follows:

- All classifiers performed better on larger training data and the accuracy leveled of around 1.6M characters.

- Both the NB and difference-in-frequency classifier showed that a larger *n*-gram improves accuracy.
- Overall, the *6*-gram NB classifier performed best. The only exception was with the 15 character window, where the SVM performed best for the 200K dataset. For this window the accuracies of the SVM and the *6*-gram NB classifier were very close.
- The *3*-gram SVM generally performed better than both the *3*-gram NB and difference-in-frequency classifier. The *3*-gram NB classifier outperformed the SVM for the smallest dataset and the two larger character windows.

A 15 character input string showed significantly higher error rates than for the larger windows. A confusion matrix analysis made it clear that samples were often classified incorrectly as other languages within the same language families. It is therefore possible to identify language families with this character window, but finer distinctions are not possible. For the larger window sizes the confusion is not that significant and classification is better.

For unseen *n*-grams in the training model a penalty was assigned in both the NB and the difference-in-frequency classifier. These penalties also have an effect on the performance accuracy. We found that for the NB model unseen *n*-grams need to be penalized more for larger character windows and sizes of *n*. The difference-in-frequency classifier showed better results when a fixed penalty score is added for an unseen *n*-gram, and for optimal parameter values performs comparably to the NB classifier.

5.3 CONTRIBUTIONS

5.3.1 LANGUAGE PROCESSING APPLICATIONS

Our software can be used as a module in a language processing system to classify documents. The system is not limited to the South African languages and new languages can be added easily. To add a language to the SVM classifier, an extra k binary classifier must be built, where k is the number of languages in the current model. For the NB classifier the process is even simpler: A new language model is created by calculating the *n*-gram statistics of the new document. Several applications of this classifier can be envisioned. For example, in

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Appendix C we describe a system that was developed to select monolingual text from an unedited corpus. Similarly, a practical text-to-speech system in a multi-lingual environment can benefit from our system by selecting the correct lexicon and pronunciation dictionary when language changes occur within the text that is to be synthesized. This same benefit applies wherever pronunciation prediction is required. Multi-lingual information retrieval is also aided greatly by a language-selection capability based on text-based LID.

5.3.2 EVALUATING AND COMPARING CLASSIFIERS

The experimental process identified and investigated several factors which are relevant to the comparison of classifiers. Classifiers were trained with various amounts of text, demonstrating how this variable influences classifier performance. The sizes of the windows used for evaluation were also varied, as was the n-gram size used for input. Interactions between these variables were studied, and suggestions for appropriate parameter choices were made. We consistently measured the amount of training data and the size of text fragments in terms of the number of characters, because other metrics make comparisons less informative. By using 10-fold cross validation throughout, we were able to obtain statistically robust results.

5.3.3 COMMON DATASET

Currently there are no common datasets for comparing text-based LID performance. In spoken LID, the OGI multilingual corpus has been greatly beneficial in comparing different systems, and we are in the process of releasing our text data for the same purpose.

5.3.4 TEXT PROCESSING

In the initial phase, when preprocessing was performed on the collected text data, a few modules were developed that can be useful in other applications. Some examples are:

- Text data from different coding schemes could be handled and converted to UTF-8 format.
- Word statistics and n-gram statistics can be calculated.
- Textual data can be extracted from HTML (Hypertext Markup Language) files.

- Sentences can be extracted from text files.

5.4 FUTURE WORK

5.4.1 DOCUMENT SIMILARITIES

In [10] the relationship between the *n*-gram statistics of two documents and the subjects of the documents were investigated. The task would probably hold different challenges than text-based LID, but the same research issues as in our study need to be addressed.

5.4.2 DETECTING CODING SCHEMES

The web contains large amounts of text and is an ideal source for text-based studies. It would be possible to initiate an intelligent web crawler for collecting specific text. Processing the collected text can be problematic because web text differs in coding schemes. A program that can detect the coding scheme of the text can become useful. The text can then be converted to the desired encoding for the application where it will be used.

5.4.3 IMPROVING TEXT-BASED LID ACCURACY

To further improve the performance of our system, several strategies may be pursued. Some possibilities include:

- Assigning weights to *n*-grams, something similar to the IDF weight.
- Training the classifiers with different features, such as:
  - The most distinctive *n*-grams of the language
  - A number of most frequent *n*-grams
  - Combining features, e.g words and *n*-grams
  - Variable length *n*-grams
  - Morphemes
  - Excluding special characters
  - Kernel strings

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

It may also be possible to reduce the size of the feature space in a systematic fashion, so that more powerful classifiers can be employed for larger values of n.

5.4.4 THEORETICAL INSIGHT INTO ALGORITHM BEHAVIOUR

The research presented in this dissertation was mostly of an empirical nature. In some cases we were able to understand the findings in an intuitive fashion, but most of our findings do not have any theoretical underpinning. We have started on an analysis of some matters (e.g. the optimal penalty value for unseen n-grams), and the development of this and other theoretical tools is an interesting challenge for future work.

REFERENCES

[1] Y.K. Muthusamy, E. Barnard and R.A. Cole, "Automatic language identification: a review/tutorial," *IEEE Signal Processing Magazine*, Vol. 11, pp. 33 - 41, October, 1994.

[2] M.A. Zissman, "Language identification using phoneme recognition and phonotactic language modeling," *Proceedings of 1995 International Conference on Acoustics, Speech and Signal Processing*, Vol. 5, pp. 3503-3506, Detroit, Michigan, May, 1995.

[3] H.P Combrinck and E.C. Botha, "Text-Based Automatic Language Identification," *Proceedings of the 6th Annual Symposium of the Pattern Recognition Association of South Africa*, Gauteng, South-Africa, November, 1995.

[4] M. Peche, M.Davel and E.Barnard, "Phonotactic spoken language identification with limited training data," *Proceedings of Interspeech 2007*, Antwerp, Belgium, To be published.

[5] W.B. Cavnar and J.M Trenkle, "N-Gram-Based Text Categorization," *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, Nevada, pp. 161-169, 1994.

[6] T. Dunning, "Statistical identification of language," Computing Research Lab, New Mexico State University, Technical Report CRL MCCS-94-273, March, 1994.

[7] G-I. Kikui, "Identifying the coding system and language of on-line documents on the internet," *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, Denmark, pp. 652-657, 1994.

[8] K. R. Beesley, "Language identifier: A computer program for automatic naturallanguage identification of online text," *Language at Crossroads: Proceedings of the 29th*

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Annual Conference of the American Translators Association, Seattle, Washington, pp. 47-54. October, 1998.

[9] G. Grefenstette, "Comparing two Language Identification Schemes," *Third International Conference on Statistical Analysis of Textual Data,* Rome, Italy, December, 1995.

[10] M. Damashek, "Gauging similarity with n-grams: Language independent categorization of text," *Science*, Vol. 267, pp. 843-848, 1995.

[11] J.M. Prager, "Linguini: language identification for multilingual documents," *Journal of Management Information Systems*, Vol. 16, No. 3, pp. 71-101, 1999.

[12] M. Padro and L. Padro, "Comparing methods for language identication," *Proceedings of the XX Congreso de la Sociedad Espanola para el Procesamiento del Lenguage Natural*, Barcelona, Spain, 2004.

[13] C. Kruengkrai, P. Srichaivattana, V. Sorlertlamvanich and H. Isahara, "Language Identification Based on String Kernels," *IEEE International Symposium on Communications and Information Technology*, Beijing, China, Vol. 2, pp. 926-929, October, 2005.

[14] P. Sibun and J.C. Reynar, "Language identification: Examining the issues," *Proceedings of the 5th Symposium on Document Analysis and Information Retrieval*, Las Vegas, Nevada, pp 125-135, 1996.

[15] C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, Vol. 2, No. 2, pp. 121-167, 1998.

[16] N. Cristianini and J. Shawe-Taylor, "*An introduction to support vector machines and other kernel-based learning methods,*" 9th ed., Cambridge University Press, 2005.

[17] L-F. Zhai, M. Siu, X. Yang and H. Gish Discriminatively trained language models using support vector machines for language identification," *IEEE Odyssey 2006: The Speaker and Language Recognition Workshop*, San Juan, Puerto Rico, pp. 1-6, June, 2006.

[18] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels," *Journal of Machine Learning Research*, Vol. 2, pp. 419-44, 2002.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

[19] D. Elworthy, " Language identification with confidence limits," *Proceedings of the 6th Annual Workshop on Very Large Corpora*, Montreal, Quebec, Canada, 1998.

[20] A. Poutsma, "Applying Monte Carlo Techniques to Language Identification," *Proceedings of Computational Linguistics in the Netherlands*, Twente, The Netherlands, 2001.

[21] A. Xafopoulos, C. Kotropoulos, G. Almpanidis and I. Pitas "Language Identification in web documents using discrete HMMs," *Journal of Pattern Recognition*, Vol. 37, pp. 583-594, 2004.

[22] A. S. House and E. P. Neuberg, "Toward automatic identification of the language of an utterance. I. Preliminary methodological considerations," *Journal of the Acoustical Society of America*, Vol. 62, No. 3, pp.708-713, 1977.

[23] A. Binas. (2005). Markovian time series models for language identification. [Online]. Available: http://www.cs.toronto.edu/ abinas/csc2515report.pdf. [Last accessed: 20/08/2007].

[24] J. Hakkinen and J. Tian, "n-Gram and Decision Tree Based Language Identification For Written Words," *IEEE Workshop on Automatic Speech Recognition and Understanding*, Trento, Italy, pp. 335-339, December, 2001.

[25] S. MacNamara, P. Cunningham, and J.Byrne, "Neural networks for language identification: A comparative study," *Information Processing and Management*, Vol. 34, No. 4, pp. 395-403, 1998.

[26] J. Tian and J. Suontausta, "Scalable Neural Network Based Language Identification from Written Text," *IEEE International Conference on Acoustic, Speech and Signal Processing*, Hong Kong, China, Vol. 1, pp. 48-51, April, 2003.

[27] K.N Murthy and G. B. Kumar, "Language Identification from small text samples," *The Journal of Quantitative Linguistics*, Vol .13, No. 1, pp. 57-80 , 2006.

[28] R. Jalam.and O. Teytaud, "Kernel-based text categorization," *Proceedings of the International Joint Conference on Neural Networks*, Washington, D.C, Vol. 3, pp. 1891-1896, 2001.

[29] C. Souter, G. Churcher, J. Hayes, J. Hughes and S. Johnson, " Natural Language Identification using Corpus-based Models," *Hermes Journal of Linguistics*, Vol. 13, pp. 183-203. 1994.

[30] E. Giguet, "Categorization according to language: A step toward combining linguistic knowledge and statistical learning," *Proceedings of the 4th International Workshop on Parsing Technologies*, Prague, Czech Republic, September, 1995.

[31] R.D. Lins and P. Goncalves, "Automatic language identification of written texts," *Proceedings of the 2004 ACM Symposium on Applied Computing*, Nicosia, Cyprus, March, 2004.

[32] G. Botha and E. Barnard, "Two approaches to gathering text corpora from the World Wide Web," *Proceedings of the 16th Annual Symposium of the Pattern Recognition Association of South Africa*, Langebaan, South-Africa, pp. 194, November, 2005.

[33] C. Chang and C. Lin (2001), LIBSVM : a library for support vector machines. [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm. [Last accessed: 30/07/2007].

[34] XLSTAT. [Online]. Available: http://www.xlstat.com/en/download/. [Last accessed: 20/08/2007].

APPENDIX A

SUPPORT VECTOR MACHINE

We will firstly describe the maximal margin classifier that works only on linearly separable data. Then, the problem of classifying data that is not linearly separable will be addressed, followed by the application of the SVM to text-based language identification.
The linear SVM determines a hyperplane that maximizes the margin between classes. If we consider a two-class binary classifier we would like the SVM to create a hyperplane which takes the form

$$\mathbf{w} \cdot \mathbf{x} - b = 0, \tag{A.1}$$

where $\mathbf{w}_j$ is perpendicular to the hyperplane. Figure A.1 demonstrates the hyperplane and support vectors trained from samples of two classes. The two parallel hyperplanes that forms the margin can be described by the equations:

$$\mathbf{w} \cdot \mathbf{x} - b = 1 \tag{A.2}$$

and

$$\mathbf{w} \cdot \mathbf{x} - b = -1. \tag{A.3}$$

The distance between the margins can be expressed as $\frac{2}{||\mathbf{x}||}$, so we want to minimize $||\mathbf{x}||$.

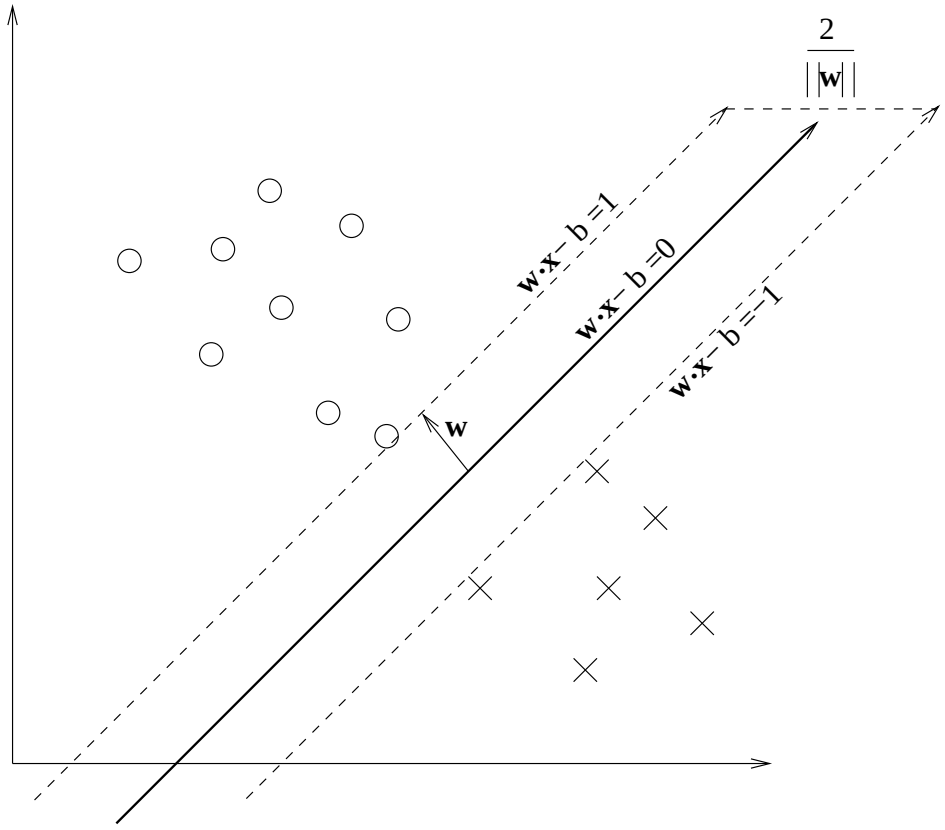

Figure A.1: A maximal margin hyperplane with its support vectors.

To prevent that data falls inside the region between the margins, we place the following constraints on $\mathbf{w}$:

$$c_i(\mathbf{w} \cdot \mathbf{x} - b) \geq 1, 1 \leq i \leq k, \tag{A.4}$$

where c_i is 1 or -1 and i is the sample number. The solution is a quadratic programming optimization problem. We want to minimize $\frac{1}{2||\mathbf{x}||^2}$, subject to $c_i(\mathbf{w} \cdot \mathbf{x} - b) \geq 1, 1 \leq i \leq k$.. Using Lagrange multipliers to solve the problem it can be shown .that the classification task is only a function of the support vector machine. Thus the training data that lie on the margins, as in Figure A.1. If the data is not linearly separable a soft margin method is used to choose a hyperplane that splits the data. By introducing a slack variable ξ, we can use the same quadratic programming optimization as with the maximal marginal classifier to solve the equation:

$$c_i(\mathbf{w} \cdot \mathbf{x} - b) \geq 1 - \xi, 1 \leq i \leq k. \tag{A.5}$$

A kernel method can be used to create non-linear classifiers. The algorithm is similar to the maximal margin classifier except that every dot product in equation A.4 is replaced by

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

a non-linear kernel function. The classifier transform the non-linear data in the input space to a high dimensional feature space where the data is (approximaltely) linearly separable. Various functional forms for the kernel have been proposed; for a function to be a kernel it needs to comply with Mecer's theorem [15]

The SVM was one of the algorithms used in our text-based LID study. Each *n*-gram can be considered as a feature in the feature space. A data point would contain the probabilities of each *n*-gram in the text string. Due to the large feature space, a kernel is nevertheless used to map the input space to more dimensions so that classes can be separable and classification is possible.

APPENDIX B

A GRAPHICAL REPRESENTATION OF LANGUAGE DISTANCES

In Chapter 4 we were able to graphically map languages from subfamilies using MDS. The maps were constructed form the confusion matrices of a 15 character window. Figures B.1-B.4 show the graphical representations if other confusion matrices are considered.

In Figure B.1 and B.2 mapping was done using the confusion matrices of the 100 character windows. For the 200K dataset, family groups can still be distinguished. However, the languages within the families are more distant from one another compared to the mapping of the 15 character representations (Figure 4.6). The languages in the families are even more separable for the larger dataset.

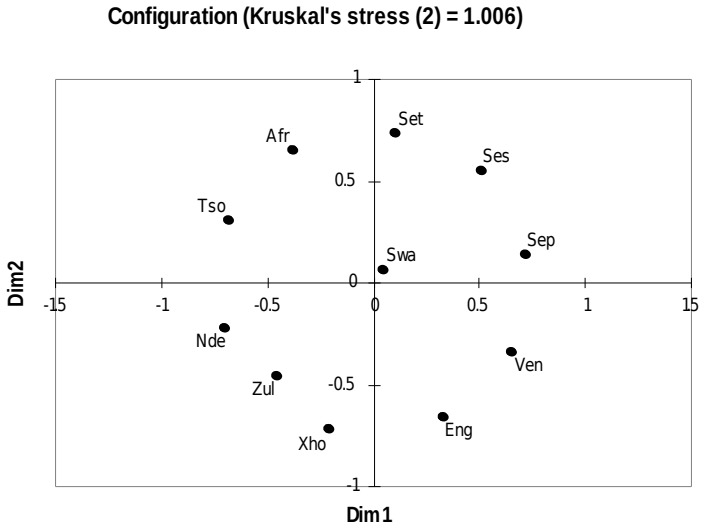

Figure B.1: Multi-dimensional scale representing similarities between languages. The similarities were calculated from the confusion matrix of the SVM classifying a 200K dataset on a 100 character window.

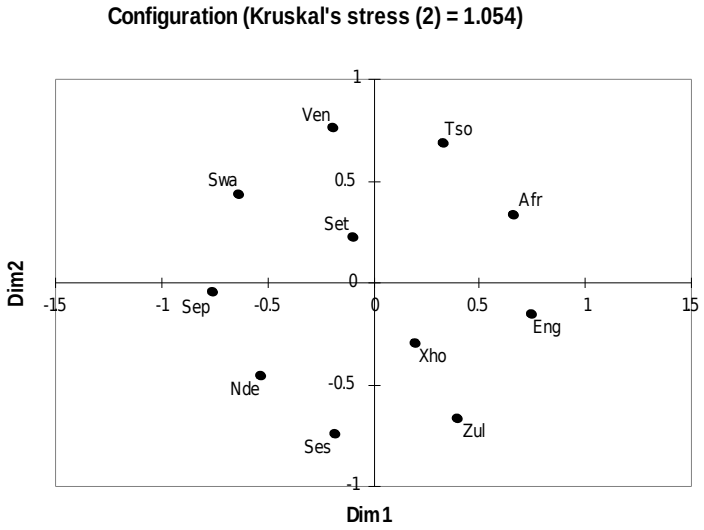

Figure B.2: Multi-dimensional scale representing similarities between languages. The similarities were calculated from the confusion matrix of the SVM classifying a 2M dataset on a 100 character window.

Figures B.3 and B.4 present the mapping using the 300 character window. For the smaller dataset language families are relatively distinguishable, but for the larger dataset the languages are scattered and less clearly grouped according family. This is not surprising: for the larger windows, there is less confusion between the languages, which makes MDS less effective (which is indicated by increased Kruskal stress values).

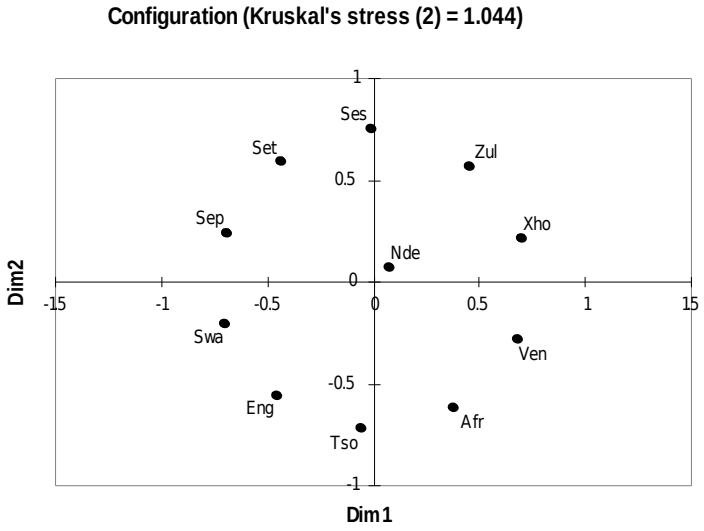

Figure B.3: Multi-dimensional scale representing similarities between languages. The similarities were calculated from the confusion matrix of the SVM classifying a 200K dataset on a 300 character window.

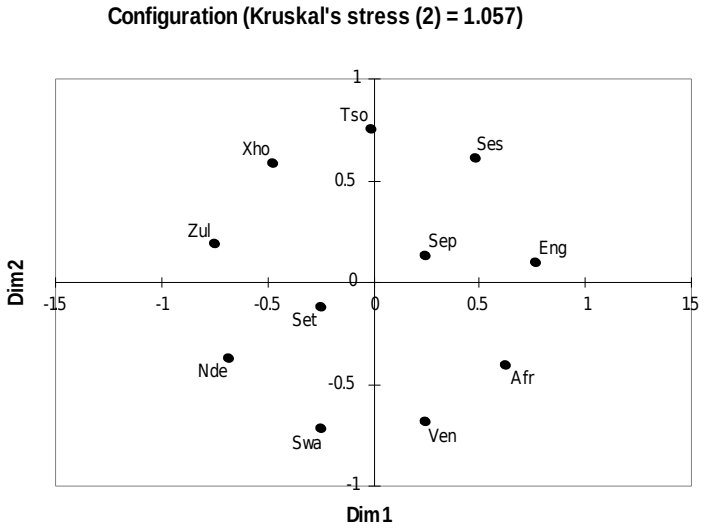

Figure B.4: Multi-dimensional scale representing similarities between languages. The similarities were calculated from the confusion matrix of the SVM classifying a 2M dataset on a 300 character window.

The general conclusion from the analysis is that the confusion between languages can be effectively combined with MDS to represent language relationships; this method is most successful when applied in environments where a relatively high proportion of utterances are confused.

APPENDIX C

APPLYING LID TO SELECT MONOLINGUAL TEXT

Many documents contain words from several languages, which can be problematic in applications where monolingual text is required. For such applications, blocks of text containing foreign words (and possibly also names and abbreviations) need to be removed before any further language processing can take place. In this experiment we investigate the use of our classifier to accomplish this task.

The NB classifier provides two potentially useful scores for the detection of foreign text, namely the likelihood (which is defined in Section 2.4.2) and the posterior probability can be expressed as

$$P(L|N) = \frac{P(L)P(N|L)}{P(N)}, \tag{C.1}$$

where $P(L|N)$ is the posterior probability (the conditional probability of the language model given the *n*-gram). $P(L)$ is the prior probability of the language model and since all language models are equally probable this term can be excluded. $P(N|L)$ is the likelihood (the conditional probability of the *n*-gram given the language model) and $P(N)$ is the prior probability of the *n*-gram.

Here, we present results when applying these measures to text which is primarily in the Afrikaans language. Using a threshold on the posterior probability, we wanted to detect unwanted abbreviations, names (e.g. place names and personal names) and foreign (mostly, English) words in sentences of the language. We refer to sentences containing the unwanted words as "contaminated sentences" and "pure sentences" do not contain any such words. For example, the string "Toyota Land Cruiser", within the Afrikaans sentence; "Die spesiale spesifikasie van die Toyota Land Cruiser met petrol sowel as diesel enjins is verbeter", causes the sentence to be classified as contaminated. It is interesting to note that 'land', 'petrol' and 'diesel' are acceptable in both Afrikaans and English.

Figure C.1 plots the posterior probability $P(L|N)$ for English and Afrikaans for the *6*-grams in the string sequence: "Die spesiale spesifikasie van die Toyota Land Cruiser met petrol sowel as diesel enjins is verbeter". Initially the probabilities are high for Afrikaans and low for English *n*-grams. From the 30^{th} *n*-gram the probabilities increase drastically for the English *n*-grams whereas the opposite takes place for Afrikaans. This is where the unwanted string starts. For the last part of the sentence the probabilities do not stabilize and for some *n*-grams the Afrikaans and English probabilities are close. This is reasonable because of the bilingual words at the end of the sentence. The graph clearly shows that it is not possible to use a threshold to distinguish the unwanted text. In Figure C.2-C.4 we calculated the moving average of the posterior probabilities for different averaging lengths. Averaging lengths of 5 and 15 produce fairly clear indications of the foreign text, but for a window of 30 the low probabilities within the region of the unwanted string are averaged out and will lead to incorrect acceptance of the sentence. The window of 15 seemed most promising, and we applied it in a test. A hundred sentences were gathered; half of these were "pure" Afrikaans sentences and the other half "contaminated" sentences. We used our classifier to discriminate between the sentences based on a posterior probability threshold. Table C.1 shows the number of wrongly classified sentences in each class. For lower thresholds, more "contaminated sentences" are misclassified than "pure sentences" and for higher thresholds more "pure sentences" are misclassified than "contaminated sentences". The best performance was surprisingly at high values of 0.55 0.65. Place names such as

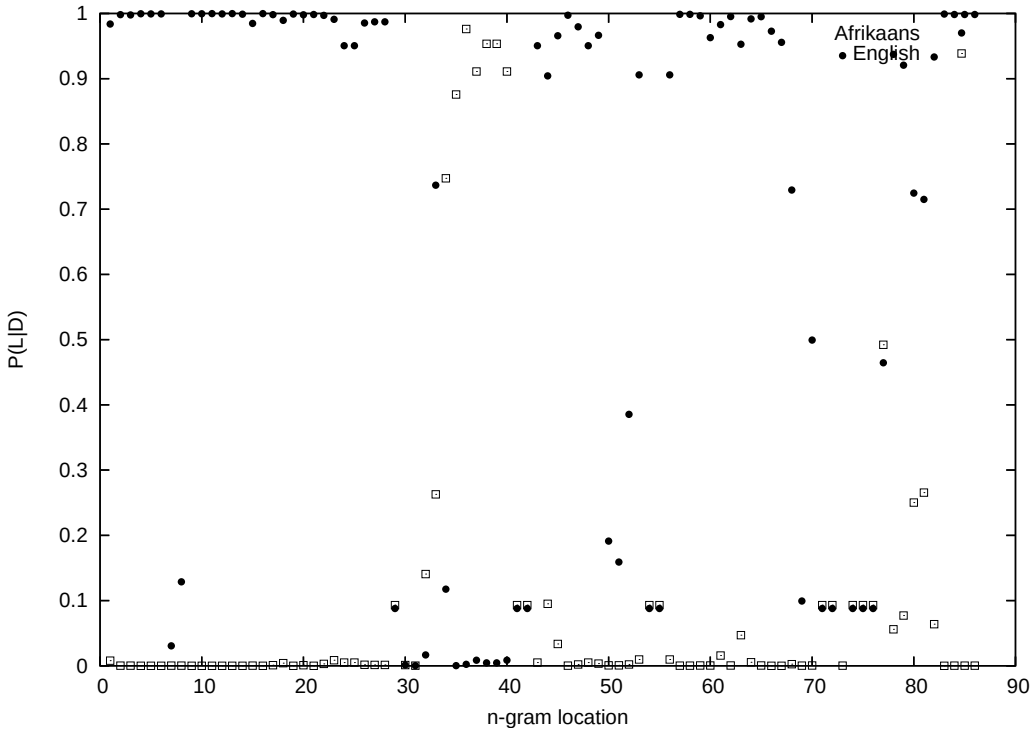

Figure C.1: The posterior probability for each *6*-gram in the character string, "Die spesifikasie van die Toyota Land Cruiser met petrol sowel as diesel enjins is verbeter".

"Bloemfontein", personal names such as "Du Preez" and small English words such as "fun" were the most difficult to distinguish. The accuracy achieved (better than 95% overall) is nevertheless highly encouraging, and we have employed this technique to filter large text corpora for a data-collection project.

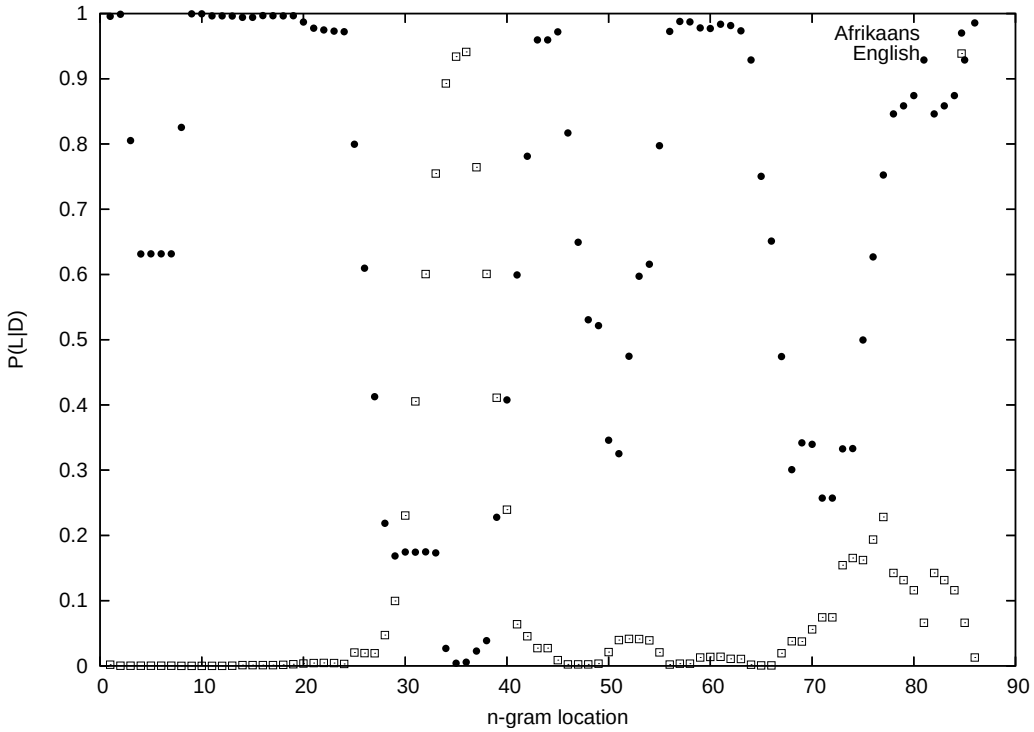

Figure C.2: The moving average of the posterior probability over a window of 5 *6*-grams.

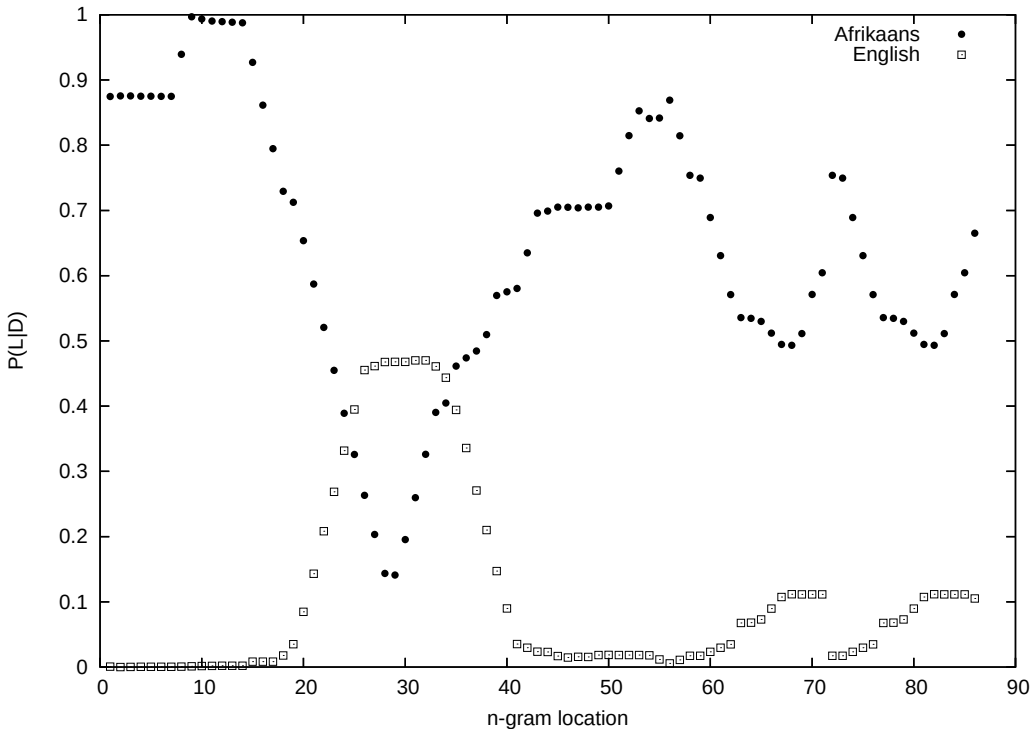

Figure C.3: The moving average of the posterior probability over a window of 15 *6*-grams.

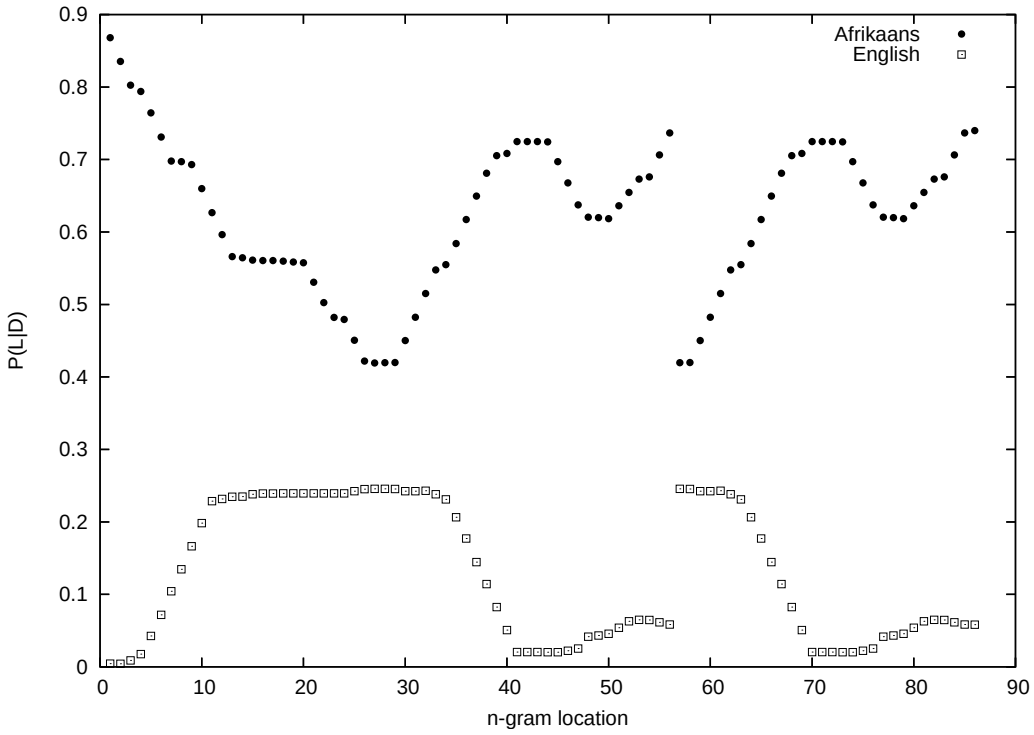

Figure C.4: The moving average of the posterior probability over a window of 30 *6*-grams.

Threshold	"pure sentences"	"contaminated sentences"
0.2	0	30
0.25	0	24
0.3	0	21
0.35	0	16
0.4	0	12
0.45	0	9
0.5	0	5
0.55	0	4
0.6	1	3
0.65	1	2
0.7	9	2
0.75	13	1
0.8	21	0

Table C.1: This table shows the number of sentences wrongly classified in each class. Fifty sentences of each class was classified. Each run was performed with a different threshold. The moving average was calculated over a window of 15 *6*-grams.