N-gram based Language Identification of Individual Words

Oluwapelumi Giwa and Marelie H. Davel
Multilingual Speech Technologies, North-West University,
Vanderbijlpark, South Africa
oluwapelumi.giwa@gmail.com, marelie.davel@gmail.com

***Abstract*—Various factors influence the accuracy with which the language of individual words can be classified using n-grams. We consider a South African text-based language identification (LID) task and experiment with two different types of n-gram classifiers: a Naïve Bayes classifier and a Support Vector Machine. Specifically, we investigate various factors that influence LID accuracy when identifying generic words (as opposed to running text) in four languages. These include: the importance of n-gram smoothing (Katz smoothing, absolute discounting and Witten-Bell smoothing) when training Naïve Bayes classifiers; the effect of training corpus size on classification accuracy; and the relationship between word length, n-gram length and classification accuracy. For the best variant of each of the two sets of algorithms, we achieve relatively comparable classification accuracies. The accuracy of the Support Vector Machine (88.16%, obtained with a Radial Basis function) is higher than that of the Naïve Bayes classifier (87.62%, obtained using Witten-Bell smoothing), but the latter result is associated with a significantly lower computational cost.**

Index Terms: text-based language identification, smoothing, character n-grams, Naïve Bayes classifier, support vector machine.

I. Introduction

Code switching is the act of mixing words from different languages within a single sentence. In running text, words occurring side-by-side originally may have come from different languages; that is, in a code-switched sentence or phrase, it is typical to find individual words from one language (referred to as the $embedded$ language) embedded within a larger sentence of a different language (referred to as the $matrix$ language) [1]. In regions with multiple languages, it is a common act amongst younger generations to use mixed language within text conversations [2]. A specific scenario is the use of numerical digits, where people often prefer using numbers from a different language to the matrix language.

In speech and text processing systems, identifying code-switched words is important for applications such as machine translation, speech synthesis, information extraction and pronunciation prediction. For example, Bhargava and Kondrak [3] showed that text-based language identification (LID) can be used to improve the accuracy of grapheme-to-phoneme conversion, and both Font Llitjos and Black [4] and Church [5] demonstrated that being able to identify the language of origin of a name (from its orthography) is important in being able to predict the possible pronunciations of that name. In related work, Basson and Davel [6] showed that the ability to identify code-switched words from orthography alone can be a useful step in building optimised grapheme-based automatic speech recognition (ASR) systems.

We approach the LID task using n-grams. N-gram based methods are widely used ([7], [8], [9]), and highly suitable for LID with small training sets and short test samples [9]. Botha and Barnard [10] applied Naïve Bayes (NB) classification and Support Vector Machines (SVMs) using n-grams to 11 South African languages, using segments of 15 to 300 characters each. While smoothing was not investigated in [10], Vatanen et al. [9] demonstrated the importance of smoothing when using n-gram models to classify short text segments. The current study extends previous contributions by comparing the two classification techniques mentioned above (SVMs and NB classifiers) for the classification of individual words rather than general text segments; and by analysing the effect of smoothing, specifically. In this context, the relationship between word length, n-gram length and classification accuracy is also investigated.

The paper is structured as follows: Section II provides an overview of different LID techniques. Section III describes the methods used during experimentation in more detail. Section IV provides an overview of the experimental approach followed and describes the data set used. Section V presents the various experiments and results. A summary of main findings concludes the paper in Section VI.

II. Background

In principle, the language origin of an input text string can be estimated by creating a model per language (from training data) and selecting the best-fitting model to predict the language source of the input string. Statistical text-based LID techniques include: Naïve Bayes classification [10], ranking methods [7], Markov models [8], [11], support vector machines [12], decision trees [13] and k-nearest neighbour classification [14]. Many LID experiments have adopted character n-gram models; such techniques have demonstrated good performances over a variety of applications.

To the best of our knowledge, limited previous work has focused on identifying the language of generic words in isolation, with more results available with regard to LID of running text. (Exceptions being [15], [13], [3], discussed below.).

When classifying longer text segments, accuracy quickly approaches 100% given enough text; for example, Cavnar $etal.$ [7] used rank difference to predict the distance between the most frequent n-gram in the language model and the test document. They extracted their evaluation set from Usenet newsgroup articles written in 14 different languages. They achieved an accuracy of 99.8% on text of 300 characters or more, while retaining the first 400 most common n-grams up to length 5. In related work, Kruengkrai $etal.$ [12] showed a similar result when classifying 17 languages with average length of 50 bytes, while ignoring character-encoding system during processing (that is, irrespective of the number of characters, 50 bytes of data were used). They achieved an accuracy of 99.7% with an SVM classifier.

Classification of a short textual fragment is more complex due to the lack of contextual information. Short text segments include proper names, generic words in isolation and very short sentences (less than approximately 15 characters). Vatanen $etal.$ [9] used the Cavnar ranking method and SVMs to identify short text segments. They experimented with 281 languages using a fairly small training set, and for test samples within the range of 5-21 characters, they obtained an accuracy of less than 90%. Similarly, Bhargava and Kondrak [3] used SVMs to classify proper names while training on a small data set of 900 names and testing on 100 names. They obtained their best identification rate of 84% using a support vector machine with a radial basis function (RBF).

Not all methods can be applied to words in isolation, with linguistic models (such as the stop words used by Johnson [16] or the closed grammatical classes used by Lins and Gonçalves [17]) not applicable to this task. One technique that is not n-gram based that is worth mentioning, is the use of a data compression model for LID, as introduced by Hategan $etal.$ [15]. They evaluated the performance of the algorithm on individual names and isolated words from 6 European languages, and reported an accuracy of above 80% on two-best results.

N-gram based method has been compared directly to other LID approaches in a number of studies. Hakkinen and Tien [13] compared a decision tree and n-gram methods. They concluded that the n-gram based method perform better on longer text samples while decision trees do better on short words like proper names. They also emphasised that the decision tree method does well with learning lexical structure information. In recent work, Baldwin and Lui [18] used SVMs and the Naïve Bayes algorithm for LID. Their experiment was carried out on 17 European languages from Wikipedia. They observed that SVMs performed better on short text segment and concluded that the shorter the text the more difficult it is to classify. Similarly, Vatanen et al. [9] experimented with two classifiers and smoothing techniques in identifying short text segments. Their reports show that Naïve Bayes classification outperformed a ranking method on sample text length in the range of 5 to 21 characters. To increase identification accuracy they test different smoothing techniques such as Katz smoothing, absolute discounting, and modified Kneser-Ney discounting. They observed the best result with absolute discounting.

Apart from the sample text length, the accuracies of these approaches depend on various other factors. Botha and Barnard [10] discussed different factors that could influence text-based LID accuracy. These factors included: size of the training data, input text size, n-gram size, LID techniques employed and language similarities.

III. Methods

In this section, we describe the n-gram based classifiers and smoothing techniques employed in further experiments.

A. LID using a Naïve Bayes classifier

A Naïve Bayes classifier uses the concept of Bayes' theorem [19]. This classifier assigns the most likely class to an input string, based on the highest a posteriori probability, given the input string. For T-LID, a Naïve Bayes classifier can be constructed using n-grams as features. Let T be a set of training samples and let each sample be represented by n feature vectors, X = $x_1, x_2, ..., x_n$, with their class labels. Let there be m classes: $K_1, K_2, ..., K_m$. To predict, a sample X is selected to belong to class K_i, if and only if:

$$P(K_i \mid X) > P(K_j \mid X); for\ 1 \leq j \leq m;\ j \neq i \qquad (1)$$

where $P(K_i \mid X)$ is the probability of a class K_i given a sample. Bayes' theorem states that:

$$P(K_i \mid X) = \frac{P(X \mid K_i)P(K_i)}{P(X)} \qquad (2)$$

where $P(X \mid K_i)$ represents the likelihood of a sample X belonging to class K_i, and $P(X)$ does not influence model comparison. The class a priori probability, $P(K_i)$, represents the count relative frequency in the sample set. According to the Naïve Bayes assumption, statistical independence of features is assumed, and the class K_i is selected such that $\prod_j P(x_j \mid K_i)P(K_i)$ is optimised, where $P(x_j|K_i)$ is then the likelihood of a specific n-gram being observed in a given language, and the word being classified consists of j n-grams.

B. Support Vector Machines

Support vector machines estimate a linear hyper-plane, which separates two binary classifiers while maximising the distance from the hyper-plane to the class samples. It was first introduced by Vapnik [20].

Data normalisation or scaling is a sensitive part of SVM training and testing. Normalisation is a way of reducing the weight of frequent n-gram counts by preventing larger n-gram counts from dominating smaller n-gram counts. Various benefits are associated to normalising data, which include speeding up training and avoiding computational complexity with numerical values. For further details on which SVM libraries were used, the size of n-gram models and how we normalised our data, see Section V-D.

C. Smoothing in the context of LID

Rare or unseen n-grams can result in poor probability estimates. 'Smoothing' refers to a range of techniques that re-distribute probability density among rare or unseen tokens [21]. As maximum likelihood estimates (MLE) are used by Naïve Bayes classifiers to estimate class probability, smoothing can help to address poor probability estimates, which result in zero probability of missing n-gram sequence models.

Different smoothing techniques have been proposed and applied to the T-LID task [21], such as: Laplace smoothing (simply adding one count across a data set), Katz smoothing [22], Witten-Bell smoothing [23], absolute discounting [24], Kneser-Ney discounting [24] and Jelinek-Mercer [25] methods. We exclude Modified Kneser-Ney (regarded as a state-of-the-art smoothing technique) from further experiments due to the small vocabulary size of this task, since Modified Kneser-Ney assumes a larger vocabulary size [21]. Rather, this work focuses on three of the above-mentioned smoothing techniques, as discussed below:

1) Katz backoff with Good-Turing discounting: In the speech recognition domain, Katz smoothing is a widely used smoothing technique [21]. It uses Good-Turing discounting to calculate adjusted counts, C^*, which determine how much probability density goes to unseen n-grams. Katz smoothing can be represented as:

$$P^*(x_i \mid x_{i-N+1}^{i-1}) = \frac{C^*(x_{i-N+1}^{i})}{\sum_{x_i} C^*(x_{i-N+1}^{i})} \tag{3}$$

where $C(x)$ counts how many times x appears in the training set, x_i represents the position of the i^{th} character in the given context, N is the n-gram parameter, and C^* is an adjusted count, with:

$$\begin{aligned} C^*(x_{i-N+1}^{i}) &= d_i C(x_{i-N+1}^{i}) \text{ if } C(x_{i-N+1}^{i}) > 0 \\ &= \alpha(x_{i-N+1})P(x_i) \text{ otherwise} \end{aligned} \tag{4}$$

where α represents a back-off weight and d_i a discount ratio according to the Good Turing estimate. This distributes leftover probability density to lower-order n-grams.

2) Witten-Bell discounting + interpolation : Witten-Bell discounting defines models recursively in terms of linear interpolation between the n^{th} and $(n-1)^{th}$ order maximum likelihood models. The discounted probability density is evenly distributed in the training set among previously unseen words with the same history. This can be represented as:

$$\begin{aligned} &P_{WB}(x_i|x_{i-N+1}^{i-1}) = \\ &\lambda_{x_{i-N+1}^{i-1}} P_{MLE}(x_i \mid x_{i-N+1}^{i-1}) \\ &+ 1 - \lambda_{x_{i-N+1}^{i-1}} P_{WB}(x_i \mid x_{i-N+2}^{i-1}) \end{aligned} \tag{5}$$

where x_i represents position of the i^{th} character in the given context, $\lambda_{x_{i-N+1}^{i-1}}$ is the discounted probability density, and $1 - \lambda_{x_{i-N+1}^{i-1}}$ is the probability mass that needs to be distributed evenly to previously unseen types.

3) Absolute discounting + interpolation: Absolute discounting uses a fixed discounting parameter, D, to reduce probability mass of seen types by subtracting a fixed value. Absolute discounting interpolates higher and lower-order n-grams by using information from lower-order n-gram models. For each seen type we subtract any fixed value between 0 and 1 from the higher-order n-gram. The estimated leftover probability mass is assigned to lower-order n-grams.

$$\begin{aligned} P_{abs}(x_i \mid x_{i-N+1}^{i-1}) = &\frac{max(C(x_{i-N+1}^{i}) \;-\; D, 0)}{\sum_{x_i} C(x_{i-N+1}^{i})} \\ &+(1 - \lambda_{x_{i-N+1}^{i-1}})P_{abs}(x_i \mid x_{i-N+2}^{i-1}) \end{aligned} \tag{6}$$

where x_i represents position of the i^{th} character in the given context, D is the discount weight, and λ is a normalising constant (that is, probability mass we have discounted).

IV. Approach

In our experiments, we aim to identify the language origin of a single word at a time. Experiments are carried out on generic text corpora from four South African languages, and results are presented in terms of LID accuracy across all languages. Using empirical measurements, we analyse the following aspects:

- For NB classifiers: the difference in classification accuracy when using word types or tokens at different training corpus sizes; and the implications of different smoothing techniques for different n-grams at different training corpus sizes.
- For SVMs: the effect of kernel choice.
- For both classifiers: the interplay between n-gram length, word length and classification accuracy.

A. Experimental Data

Generic text in three South African languages (Afrikaans, Sesotho, and isiZulu) were obtained from a pre-release of the NCHLT text corpora [26]. These corpora were collected from "gov.za" which is a South African government domain. This domain is devoted to topics relevant to South Africa, which include: news events across South Africa or beyond, speeches, and statements from various cabinet meetings in the Republic of South Africa. All text are encoded in UTF-8 format to accommodate special characters found in Afrikaans and Sesotho. Our South African English data was obtained from a broadcast news corpus [27]. We performed text normalisation to remove punctuation marks, numbers, formulae, dashes and brackets.

The data is randomly partitioned into a training set (80%), development set (10%) and test set (10%) based on the number of characters in running text. Table I contains the original data set, while Table II contains the newly partitioned set after removing overlapping words from the test and development set: that is, only unique words are retained. For the purpose of this experiment, we will refer to Tables I and II as the "all" and "unique" data sets, respectively. In Tables I and II, "*A*" represents Afrikaans, "*E*" English, "*S*" Sesotho, and "*Z*" isiZulu. The rationale behind the experiment is that the text-based LID (on running text) results are typically obtained by retaining multiple copies of the same token; word-based

TABLE I

Original data that contain all words. The number of unique words, total number of characters and average word length per language are shown.

	Total characters	Unique words	Average word length
Training Set	A - 1 840 494 E - 1 840 547 S - 1 840 697 Z - 1 840 570	A - 15 727 E - 15 765 S - 15 680 Z - 15 799	A - 13.64 E - 13.21 S - 13.80 Z - 13.38
Total	7 362 308	62 971	
Development Set	A - 221 040 E - 221 023 S - 221 087 Z - 220 933	A - 2 101 E - 2 060 S - 2 073 Z - 2 114	A - 13.28 E - 13.86 S - 12.78 Z - 12.87
Total	884 083	8 348	
Test Set	A - 221 140 E - 221 125 S - 221 007 Z - 221 041	A - 2 110 E - 2 080 S - 2 120 Z - 2 111	A - 13.28 E - 13.86 S - 13.18 Z - 13.80
Total	884 313	8 421	

TABLE II

Repartitioned data set after removing repeated words. The number of unique words, total number of characters and average word length per language are shown.

	Total characters	Unique words	Average word length
Training Set	A - 204 456 E - 204 751 S - 204 764 Z - 204 925	A - 15 727 E - 15 765 S - 15 680 Z - 15 799	A - 13.00 E - 12.99 S - 13.06 Z - 12.97
Total	818 896	62 971	
Development	A - 9 911 E - 9 931 S - 9 940 Z - 9 933	A - 740 E - 717 S - 738 Z - 710	A - 13.39 E - 13.85 S - 13.49 Z - 13.99
Total	39 715	2 905	
Test Set	A - 9 901 E - 9 961 S - 9 910 Z - 9 935	A - 744 E - 723 S - 745 Z - 707	A - 13.30 E - 13.78 S - 13.30 Z - 14.05
Total	39 707	2 919	

LID results are typically obtained on dictionaries, where only unique tokens are retained automatically.

In order to investigate the relationship between identification accuracy and training-set sizes; we conduct a series of experiments, which involves creating different training subsets of different sizes from the "all" data set, namely: 250KB, 500KB, 1M, and 1.8M. To avoid bias of having the same words that run across both the test set and train set; we construct different data subsets by removing the identical words from our data sets. Finally, our respective unique words extracted for training are 75KB, 113KB, 162KB, and 204KB in size, which are equivalent to our respective subset sizes; that is, 250KB, 500KB, 1M, and 1.8M. Table II shows our largest unique data set (240KB) together with development and test data set. We use the $test$ and $development$ set from Table II across all experiments.

B. General discussion of our setup

In order to identify the language origin of individual words, we generate text of different n-gram character lengths from our corpus. All generic words include word boundary markers (indicated by #); while we do not show the results here, we observed that using word boundary markers improve classification accuracy in all experiments. For each word all n-grams are extracted, for example, the word #BREAD# would be represented by the tri-grams #BR, BRE, REA, EAD, AD#.

During NB training, an n-gram model is trained, estimating the probabilities of the individual n-grams based on their counts in the training corpus. For example, when training a tri-gram model, a vector of tri-gram values is created in a T-dimensional space, where T is the number of possible tri-grams, and each tri-gram in the vector is associated with a specific likelihood estimate. For SVM training, a similar T-dimensional vector is constructed, but now each training sample is represented by the number of times a specific n-gram occurs in that training sample. These T-dimensional training samples are then used to train an SVM.

C. Evaluation

To evaluate the performance of the models, we evaluate their accuracies based on certain criteria. We measure LID accuracy as the proportion of correctly identified tokens in the test set, compared to the total number of tokens in the test set. Each test is characterised by the following parameters: language, training data used (size, unique/all), adopted n-gram model and technique employed.

V. Experiments

This section focuses on evaluating the accuracy achieved by the different n-gram models on the test set. We examine how each n-gram model performs with different word lengths and LID classifiers. We also investigate the influence of different training data set sizes on accuracy. Our n-gram models range from 1 to 7.

A. Naïve Bayes Baseline Back-off

Our baseline model is based on a back-off technique that estimates word probability for tokens with n-gram length greater than or equal to the word length. Using standard Naïve Bayes, the probability of an n-gram with length greater than or equal to the word length becomes zero. We need some form of back-off to be able to produce results over all test types regardless of word length or n-gram parameter (prior to investigating more sophisticated smoothing techniques).

Our baseline strategy adopts a technique that backs off to an n-gram size of the word length (W), minus a fixed value (x), where x is a small value. To obtain a suitable estimate for our fixed value, x, we performed 5-fold cross validation on our training data. We evaluate the effect of subtracting a fixed value, x, by setting x in the range of $W/2$ to W, where W represents total character counts per word without word boundaries.

For our baseline model, we find $W/2$ as the most suitable parameter for x . This back-off technique is used in our baseline model in order for us to be able to compare with more advanced techniques in later experiments.

TABLE III

Classification accuracy of baseline Naïve Bayes systems trained on unique types at different training sizes and evaluated on test set

	1-gram	2-gram	3-gram	4-gram	5-gram	6-gram	7-gram
250K	58.715	75.176	78.488	78.488	77.384	76.614	75.878
500K	58.884	73.771	78.555	78.554	77.852	77.016	76.447
1M	58.983	75.845	79.793	80.462	79.893	79.258	79.090
1.8M	58.983	76.179	80.261	80.763	80.027	79.425	79.190

B. *Unique words vs. All words*

In our first experiment, we examine the influence of using unique words (types) against all words (tokens). A reduction in computational complexity and training time are some of the benefits of using types over tokens. We therefore examine the influence of using only types against tokens in training, using the data set from Table I, partitioned into different sized subsets. For proper comparison, we use only one test data set across all different subsets.

Figure 1 shows the classification accuracies obtained between "unique" and "all", as defined above. It also shows the relative difference in identification accuracy using different n-gram models across various training data sizes. This baseline result is based on the Naïve Bayes baseline back-off described in Section V-A. Most notable, smaller n-gram models benefit most from the use of types while higher-order models perform better under all tokens, with less than 1% percentage gain over its lower-order models. These accuracies can be considered to be corpus-dependent. For further experiments, we use types over tokens (Table II), as this results in reduced training time and computational complexity.

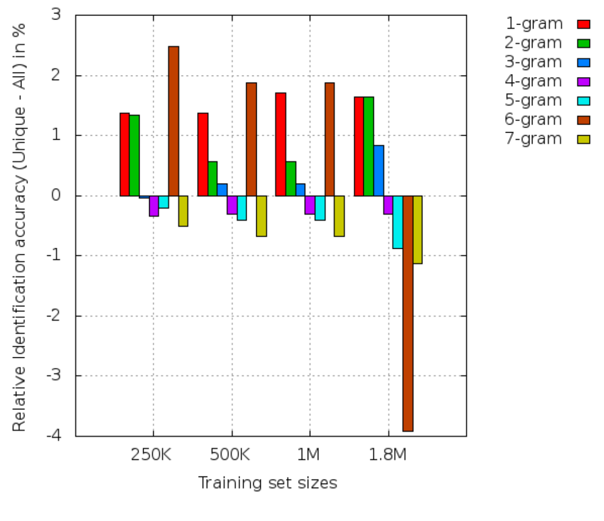

Fig. 1. Difference in LID accuracy when comparing the baseline n-gram models trained using only unique tokens with one trained on all tokens. Results are provided at different training set sizes.

Table III shows the overall identification accuracy of the Naïve Bayes Baseline back-off method across various data sizes. With smaller data sets, higher-order n-gram models show poor performance, with 7-gram model producing the worst performance. With more training data we observe an improvement in accuracy across all models; however, we notice larger improvement on lower-order n-gram models with 4-gram model benefiting most.

Note that classification accuracy increased for all models with more training data. This means that we were unable to achieve asymptotic performance with the data sets available, which implies that higher accuracies are possible with increased training data.

C. *Smoothing Analysis*

Tables IV, V, and VI show classification accuracies of three smoothing techniques using different n-gram models and training data sets. The question is how we handle unseen features in our test data. Smoothing methods help to better deal with data sparseness by borrowing probability mass from higher-order n-grams and redistributing it among lower-order n-grams.

We estimate the discount parameter value (D) for absolute discounting by applying 5-fold cross-validation on the training set. To avoid overfitting and reduction in likelihood of the held-out data, we opt for a discount value that keeps the likelihood of the held-out data stable across the validation set.

As expected, with more training data, performance increases with n-gram length. Also, we observe a reduction in classification accuracy from the 6-gram model upwards. This is probably due to both the average word length of our data set, and increased data sparsity of higher order models.

In conclusion, for the current data set, the 4-gram model seems to be the preferred model across all three modelling techniques employed. The best accuracy (of 87.72%) was obtained using Witten-Bell discounting. This outperformed both absolute discounting and Katz smoothing, even though the difference in accuracies are relatively small. In general (for higher-order models) we see an 8% accuracy increase, when comparing Witten-Bell discounting to the same n-gram model without smoothing.

TABLE IV

Classification accuracy using Witten-Bell smoothing at different n-gram lengths, evaluated on test set.

	1-gram	2-gram	3-gram	4-gram	5-gram	6-gram	7-gram
250K	72.767	83.239	85.212	85.781	85.279	84.811	84.778
500K	72.968	82.971	84.978	85.781	85.480	84.978	84.711
1M	72.700	83.673	86.417	86.250	86.785	86.283	86.183
1.8M	73.202	84.008	86.584	87.722	87.387	87.554	87.287

TABLE V

Classification accuracy using Katz smoothing at different n-gram lengths, evaluated on test set.

	1-gram	2-gram	3-gram	4-gram	5-gram	6-gram	7-gram
250K	72.533	82.937	84.343	85.212	84.778	84.744	84.744
500K	72.968	83.272	85.012	85.614	85.882	85.547	85.547
1M	64.604	82.168	85.079	84.778	84.911	85.212	85.212
1.8M	73.202	84.108	86.751	87.487	86.986	86.818	86.818

TABLE VII
LID accuracy using a Linear Kernel at different n-gram lengths, evaluated on the test set.

	2-gram	3-gram	4-gram	5-gram
250K	83.506	85.012	84.744	77.651
500K	83.473	85.547	84.778	83.473
1M	84.610	86.718	86.317	84.778
1.8M	84.744	87.454	86.986	83.674

TABLE VIII
LID accuracy using an RBF kernel at different n-gram lengths, evaluated on the test set.

	2-gram	3-gram	4-gram	5-gram
250K	85.012	86.685	85.012	81.632
500K	85.982	86.283	85.881	82.335
1M	86.952	87.086	87.621	84.744
1.8M	87.789	88.123	88.157	84.376

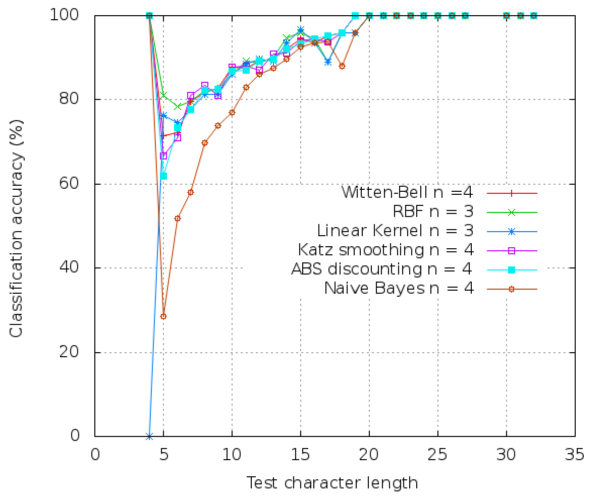

Fig. 2. LID accuracy of words of different lengths, when training with 1.8M data set, evaluated on test set.

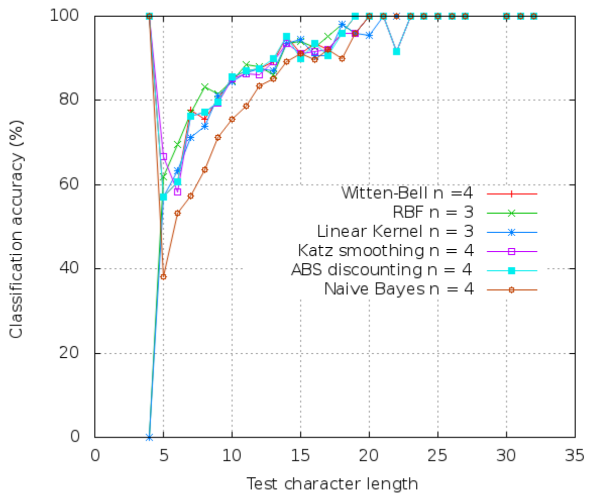

Fig. 3. LID accuracy of words of different lengths, when training with 250KB dat set, evaluated on test set.

TABLE VI
Classification accuracy using Absolute Discounting (d = 0.24) at different n-gram lengths. Accuracy evaluated on test set while d calculated by applying 5-fold cross-validation on training set.

	1-gram	2-gram	3-gram	4-gram	5-gram	6-gram	7-gram
250K	72.800	83.406	85.112	85.815	84.844	84.410	84.041
500K	72.633	83.205	84.978	85.313	85.346	84.543	84.343
1M	72.432	83.607	86.517	86.183	86.417	85.982	85.647
1.8M	72.968	84.008	86.484	87.354	86.685	86.618	85.714

D. Support Vector Machines

Using SVMs for classification involves four decisions vital to proper classification: (1) data preparation, which concerns selection of the training data and deciding how many classes should be included; (2) converting samples to feature vectors, where each sample in a training set should contain a class label, feature index and feature value; (3) data normalisation or scaling, in order to prevent over-fitting (which results in a zero mean and unit variance); and (4) proper choice of SVM kernels for good classification accuracy. Determining the kernel of choice also involves setting various hyper-parameters associated with a specific kernel. Predicting ideal hyper-parameter values can be achieved using a grid search on the development set.

This experiment employed two SVM libraries namely: LibSVM [28] and LIBLINEAR [29] for classification. These two libraries use different methods for multi-class classification. LIBSVM uses one-against-one method for classification, where one SVM is constructed for each pair of classes, classification is performed using a voting strategy. LIBLINEAR uses one-against-rest, where a classifier is trained for each class, prior to voting.

We carried out our experiment with two widely used kernels namely: a Radial Basis Function (RBF) and a linear kernel. The SVM training set was generated by combining n-grams across languages to create the SVM training set. Each selected model has a feature dimension equal to the number of n-gram combinations. After creating our feature vectors, we scaled each attribute in the range of [0, 1] in our training set. The same scaled value used in building our model was used to scale the test and development set. We performed 5-fold cross validation on our training set to obtain optimal kernel parameters. We limit our n-gram model length to 5-gram due to the longer training time and extensive resource usage associated with higher-order n-gram models.

Tables VII and VIII show classification accuracies obtained using the RBF kernel and linear kernel (respectively) and different n-gram models across various training data sets. All models improved with more training data, and the RBF kernel produced higher accuracies than the linear kernel.

E. Effect of various corpus sizes on LID accuracies

The size of the training data set is one of the factors that influences LID accuracy of running text. For the task at hand, what effect does different training data sizes have on accuracy,

TABLE IX
A comparison of LID accuracy for different classifiers investigated.

	NB (n=4)	WB (n=4)	Katz (n=4)	ABS (n=4)	Linear (n=3)	RBF (n=3)
250K	78.488	85.781	85.212	85.815	85.012	86.685
500K	78.555	85.781	85.614	85.313	85.547	86.283
1M	80.462	86.250	84.778	86.183	86.718	87.086
1.8M	80.763	87.722	87.488	87.354	87.454	88.123

and how do different classifiers and smoothing techniques perform as training data sizes increase?

In Table IX, results are shown for different LID techniques and training set sizes. On the smallest data set, NB with smoothing techniques outperformed the SVM with a linear kernel. For all training set sizes investigated, an SVM with an RBF kernal performed best. NB classification with absolute discounting and Witten-Bell smoothing, as well as SVM classification with a linear kernel, shared similar performance across a range of training data sizes.

F. *Effect of word length on classification accuracy*

In this section, we want to analyse another factor that influences classification accuracy. We are interested to see how classifiers and smoothing techniques perform at different word lengths. Using our largest and smallest training data sets, figures 2 and 3 show the classification accuracy achieved when evaluating words with different character lengths.

The fluctuation in classification accuracy for words shorter than 5 characters could be due to the fact that these words occur fairly infrequently in the test data set. The SVM RBF classifier obtained the highest performance for short character lengths (5 characters or less), while Naïve Bayes classification achieved the worst performance for the same scenario. With longer words (20 characters or more), the Naïve Bayes classifier achieves a classification accuracy of 100%. This result confirms previous results on using Naïve Bayes for long sentences. Also, with 12 or more characters, the difference in classification using Naïve Bayes with and without smoothing becomes insignificant.

VI. Conclusion

This paper examined two different classification methods which can be applied to LID of generic words in isolation. The analysis showed that high LID accuracy can be achieved on words in isolation using both methods.

We investigated the difference between using unique types and all tokens when training a Naïve Bayes classifier, and found a computational win when using unique types (that was not offset by any loss in LID accuracy). As our baseline classifier, we used a Naïve Bayes back-off method for words with character length greater than or equal to the n-gram model parameter. This method produced good results across all n-gram lengths.

The Naïve Bayes classifier achieved an identification accuracy of 80.76% using a 4-gram model when tested on unique types. To reduce the total number of unseen tokens as the n-gram model increases, we experimented with Katz smoothing, Absolute Discounting and Witten-Bell smoothing. Amongst the three smoothing techniques, Witten-Bell smoothing achieved the best performance, but only with a small margin (less than 1%). Smoothing improved average classification accuracy for higher order n-grams by a percentage of approximately 8% (from 79.89% to 87.55%).

The highest classification accuracy of 88.12% was obtained using an SVM with an RBF kernel and an n-gram length of $n = 3$. This classifier clearly outperformed the Naïve Bayes classifier without smoothing, with NB classification accuracy improving considerably when smoothing is used.

To conclude, our experiments show that better identification can still be achieved with more training data. Achieving an asymptotic performance depends not only on the available data set; it also depends on the word length, with longer words achieving very high accuracies on the 1.8M data set. For the four language task studied, the accuracy of the Support Vector Machine (88.16%, obtained with a Radial Basis function) was higher than that of the Naïve Bayes classifier (87.62%, obtained using Witten-Bell smoothing), but the latter result was associated with a significantly lower computational cost. As the computational cost increases as the training set size increases, both techniques will be usable in future work, based on the trade-off between accuracy and computational cost required for a specific application.

VII. Acknowledgement

We are grateful to Martin Puttkammer of North-West University and Thomas Nielser of Stellenbosch University, who provided us with the text data used in this analysis.

References

[1] P. Li, "Spoken word recognition of code-switched words by Chinese-English bilinguals," *Journal of memory and language*, vol. 35, pp. 757–774, 1996.

[2] P. Auer, *Code-switching in conversation: Language, interaction and identity*. Routledge, 2002.

[3] A. Bhargava and G. Kondrak, "Language identification of names with SVMs," in *Proc. NAACL-HLT*, 2010, pp. 693–696.

[4] A. F. Llitjos and A. W. Black, "Knowledge of language origin improves pronunciation accuracy of proper names." in *Proc. INTERSPEECH*, 2001, pp. 1919–1922.

[5] K. Church, "Stress assignment in letter to sound rules for speech synthesis," in *Proc. ACL*, 1985, pp. 246–253.

[6] W. Basson and M. H. Davel, "Category-based phoneme-to-grapheme transliteration," in *Proc. INTERSPEECH*, 2013, pp. 1956–1960.

[7] W. B. Cavnar, J. M. Trenkle *et al.*, "N-gram-based text categorization," *Ann Arbor MI*, vol. 48113, no. 2, pp. 161–175, 1994.

[8] T. Dunning, *Statistical identification of language*. Computing Research Laboratory, New Mexico State University, 1994.

[9] T. Vatanen, J. J. Väyrynen, and S. Virpioja, "Language identification of short text segments with N-gram models." in *Proc. LREC*, 2010, pp. 3423–3430.

[10] G. R. Botha and E. Barnard, "Factors that affect the accuracy of text-based language identification," *Computer Speech & Language*, vol. 26, no. 5, pp. 307–320, 2012.

[11] A. Xafopoulos, C. Kotropoulos, G. Almpanidis, and I. Pitas, "Language identification in web documents using discrete HMMs," *Pattern recognition*, vol. 37, no. 3, pp. 583–594, 2004.

[12] C. Kruengkrai, P. Srichaivattana, V. Sornlertlamvanich, and H. Isahara, "Language identification based on string kernels," in *Proc. ISCIT*, 2005, pp. 926–929.

[13] J. Hakkinen and J. Tian, "N-gram and decision tree based language identification for written words," in *Proc. ASRU*, 2001, pp. 335–338.

[14] P. Sibun and J. C. Reynar, "Language identification: Examining the issues," in *Proc. SDAIR*, 1996, pp. 125–135.

[15] A. Hategan, B. Barliga, and I. Tabus, "Language identification of individual words in a multilingual automatic speech recognition system," in *Proc. ICASSP*, 2009, pp. 4357–4360.

[16] S. Johnson, "Solving the problem of language recognition," Technical report, School of Computer Studies, University of Leeds, Tech. Rep., 1993.

[17] R. D. Lins and P. Gonçalves, "Automatic language identification of written texts," in *Proc. ACM symposium on Applied computing*, 2004, pp. 1128–1133.

[18] T. Baldwin and M. Lui, "Language identification: The long and the short of the matter," in *Proc. ACL*, 2010, pp. 229–237.

[19] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[20] V. N. Vapnik, *Statistical learning theory*, ser. Adaptive and learning systems for signal processing, communications, and control. Wiley-Interscience, 1998.

[21] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proc. ACL*, 1996, pp. 310–318.

[22] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," in *Proc. ICASSP*, 1987, pp. 400–401.

[23] A. M. M. Hasan, S. Islam, and M. A. Rahman, "A comparative study of Witten Bell and Kneser-Ney smoothing methods for statistical machine translation," *Journal of Information Technology*, vol. 1, pp. 1–6, 2012.

[24] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependences in stochastic language modelling," *Computer Speech & Language*, vol. 8, no. 1, pp. 1–38, 1994.

[25] F. Jelinek, "Up from trigrams," in *Proc. EUROSPEECH*, 1991, pp. 1037–1040.

[26] N. J. De Vries, J. Badenhorst, M. H. Davel, E. Barnard, and A. De Waal, "Woefzela-An open-source platform for ASR data collection in the developing world," in *Proc. INTERSPEECH*, 2011, pp. 3177–3180.

[27] H. Kamper, F. De Wet, T. Hain, and T. Niesler, "Resource development and experiments in automatic SA broadcast news transcription," in *Proc. SLTU*, 2012.

[28] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[29] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.