Text-based Language Identification for the South African Languages

Gerrit Botha, Victor Zimu, Etienne Barnard

Human Language Technologies Research Group
Meraka Institute, Pretoria, South Africa
{gbotha, vzimu, ebarnard@meraka.co.za}

Abstract

We investigate the performance of text-based language identification systems on the 11 official languages of South Africa, when *n*-gram statistics are used as features for classification. In particular, we compare support vector machines (SVMs) and likelihood-based classifiers on different amounts of input text, both from a closed domain and an open domain. With as few as 15 words of input text, reliable language identification is possible. Although the SVM is generally more accurate a classifier, the additional computational complexity of training this classifier may not be justified in light of the importance of using a large value for *n*.

1. Introduction

In a multilingual environment, language processing is often initiated with some form of language identification. For example, a general-purpose telephonic help line may require identification of spoken language before routing calls to an appropriate operator [1]. Similarly, a document-processing system may need to determine the language of textual contents before performing tasks such as topic identification, stemming, for search purposes, and such tasks. A text-to-speech system may need to determine the language of short pieces of text to determine appropriate pronunciations rules, prosodic models, and phrasing strategies.

In the current contribution, we investigate text-based language identification for the official languages of South Africa (Afrikaans, English, isiNdebele, isiXhosa, isiZulu, Sepedi, Sesotho, Setswana, siSwati, tshiVenda and xiTsonga). The general topic of text-based language identification has been studied extensively, and a spectrum of approaches have been proposed, with the most important distinguishing factor being the *depth of linguistic processing* that is utilized. At the one extreme are approaches that attempt to do a complete parse of text in order to determine not only the language spoken, but also the syntactic structure of the textual fragment. Whenever a successful parse is found, these techniques 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 language by using simple statistical measures of the text under consideration for example letter frequencies, the presence of certain keywords, etc. Conventional algorithms from pattern recognition are then used to perform language identification based on these statistics. Such techniques are the focus of the current research.

In particular, we are interested in methods that employ the frequencies of adjacent groups of *n* letters as input features. These *n*-grams are extremely simple to compute for any given text, allow for a straightforward trade-off between accuracy and complexity (through the adjustment of *n*), and have been shown to perform well in language identification and related tasks in several languages.

The central aim of the current study is to investigate the accuracy that can be achieved for the South African languages using *n*-gram statistics. This accuracy depends on a number of factors, including the following:

- *The size of the textual fragment used for language identification* - more text will certainly lead to higher accuracy, but in many applications it is desirable to identify the language spoken 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; 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 similarity of the languages to be distinguished* - the official South African languages can be grouped into a number of language families; specifically, Nguni (consisting of, isiZulu, isiNdebele, isiXhosa and seSwati), Sotho (consisting of, Sesotho, Setswana and Sepedi) and a pair that falls outside these families (xiTsonga and tshiVenda). It is predictably much harder to distinguish between languages that fall within a single family.

We investigate these factors by constructing a number of classifiers based on *n*-gram statistics for the official languages of South Africa. In Section 2 we give a more detailed description of the data that was used for training and evaluation of our system. Section 3 describes the details of our training and evaluation processes, and Section 4 contains our experimental results.

2. Data

As discussed above, we restrict our attention to the South African languages in particular, the eleven official languages of South Africa. Two data sets were used in our research: a limited-domain set (consisting of the text of the South African national constitution in all eleven languages), and an open-domain set which was created by an extension of the Web-crawling approach described in [2]. That method employed an early language-identification system for automatic selection of Web pages, and turned out to suffer from two limitations, namely wrongly identified web pages and web pages with mixed text (i.e. containing text from more than one language). We therefore had to process those pages manually to create our open-domain set, as we describe below. (For concreteness, we refer to the creation of the isiZulu corpus; similar steps were taken for the other languages as well.)

Pages are wrongly identified for a number of reasons. The first reason is that a great part of the internet is in English; as a result most of the wrongfully identified pages were English web-pages [2]. The other languages identified were isiNdebele, isiXhosa, Sesotho, siSwati and xiT-songa. The reason in this case is the closeness of orthography in various South African languages. This is more so for languages that are related to each another than others. For isiZulu these are the languages falling within the Nguni language group (i.e. isiNdebele, isiXhosa and Siswati).

Some of the pages contained a mixture of languages, i.e. isiZulu and one or more extra South African language(s). The text from the other languages was deleted and only isiZulu text retained. Through these two processes we arrived at a reliable set of isiZulu documents. In order to ensure that our open-domain set overlaps minimally with the closed-domain set, we subsequently removed all government-related documents from this collection.

3. Implementation

3.1. Constructing features

The *n*-gram method used for language identification has the advantage that no linguistic knowledge needs to be gathered to build a classifier. However, a large number of features may need to be calculated. By pre-processing these n-grams, refined features can be achieved and classifier error rate can be reduced. Experiments with such techniques and successful results were reported in [3][4][5]. The number of possible *n*-gram combinations depends on (a) the value of *n* and (b) the number of distinct "characters" contained in the orthography employed. In our tests, only letters and spaces were used in constructing a *n*-gram; upper and lower case were not distinguished. The use of punctuation marks and special characters bound to a language were not considered in the construction. Two of the South-African languages (Sepedi, Setswana) contain the special character "š". The character is much more common in Sepedi text than in Setswana, and would therefore be a useful language marker. However, this character is not marked consistently in Web-derived documents, and we have therefore mapped it to "s" in the current work.

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. 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.

3.2. Likelihood based Classifier

The likelihood based classifier creates a model that contains the frequency count of the unique *n*-grams found in training text. To classify the test text a model is build on the same principle. The training model is then normalized and the difference in *n*-gram frequencies between test and training models are calculated. The test language with the smallest difference is then classified as the likeliest language. The algorithm was implemented in the Java programming language.

3.3. Support Vector Machine

The support vector machine is a non-linear discriminant function that is able to do classification in a high dimensional space. Each *n*-gram is used as a feature for the classifier. Each sample would be a vector that contains the frequency count of each *n*-gram. The feature space grows exponentially with the size of *n* and this leads to extensive hours of training and the other factors mentioned in Section 3.1. For reasonable times to train, *n*=3 was chosen. The Weka Java API [6] provides platform with methods that implements functions to build and evaluate various classifiers. A Weka compatible SVM library [7] was used as the classifier in the experiments

3.4. Classification procedure

For the likelihood classifier the *n*=3 and *n*=6 was evaluated. For the SVM only *n*=3 was used. The number of words per sample (that is, the number of words used to count the *n*-grams that are used for classification) were also tested for different values. Thus, if we train the SVM with an *n* of 3 and 15 words, the test set also needs to be constructed in the same manner.

Both the classifiers were trained and tested with the same text, assuring no bias for the one or the other. The training set contained 200 samples per language (thus, 2 200 samples in total) and the test set contained 100 samples per language.

4. Results

4.1. In-domain tests

The error rates obtained with various window sizes and *n*=3 are shown in Figure 1 for both the SVM and the likelihood-based classifiers. For all these tests, both the training and test data were obtained from the government domain. All results are obtained with 10-fold cross validation. We see that the SVM is slightly worse than the likelihood-based classifier for a window size of 2 words, but significantly more accurate when the window size grows to 50 words. When 50 words are available, SVM is virtually flawless at this task – an error rate of 0.3% is achieved. Even for a window size of 2 words, classification is significantly better than chance (which would give an error rate of around 91% on this task). In fact, for some applications such a classifier would be quite usable, especially since the errors tend to fall within the correct language family. This phenomenon is demonstrated in the table of Figure 2, which shows the confusion matrix obtained with the likelihood-based classifier. The rows in table correspond with the true class of the input text, whereas the columns reflect the choice made by the classifier. We see that the two most confusable language pairs are (isiZulu, isiNdebele) and (Setswana, Sepedi). Since these both pairs are closely related languages, it may not be harmful for further linguistic processing if one is confused with the other on a short sample. (Despite appearances, the tendency of the classifier to choose isiNdebele and Afrikaans classes does not reflect any linguistic preference of the authors!)

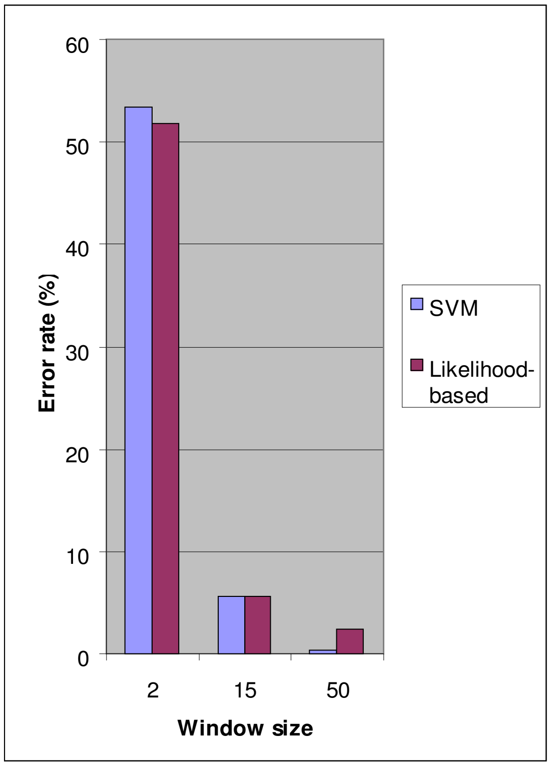

Figure 1: *Error rates of two classifiers, for n=3*

Afrikaans	English	IsiNdebele	Sepedi	Sesotho	SiSwati	XiTsonga	Setswana	Tshivenda	IsiXhosa	IsiZulu	
97	3	0	0	0	0	0	0	0	0	0	**Afrikaans**
22	**78**	0	0	0	0	0	0	0	0	0	**English**
6	1	**86**	0	0		1	0	1	4	1	**IsiNdebele**
26	11	25	**35**	1	0	0	2	0	0	0	**Sepedi**
19	3	17	29	**32**	0	0	0	0	0	0	**Sesotho**
4	0	23	0	0	**65**	0	0	0	6	2	**SiSwati**
10	0	28	3	3	13	**36**	0	2	5	0	**XiTsonga**
17	1	18	48	5	1	1	**9**	0	0	0	**Setswana**
14	8	26	3	0	2	8	0	**39**	0	0	**Tshivenda**
6	0	43	0	0	5	0	0	0	**46**	0	**IsiXhosa**
7	0	56	0	0	9	0	0	3	18	**7**	**IsiZulu**

Figure 2: *Confusion matrix obtained with likelihood-based classifier, for n=3 and a window size of two words*

To evaluate the effect of *n*, we have constructed likelihood-based classifiers for *n*=3 and *n*=6. Figure 3 shows that the larger window size consistently outperforms the smaller window, and achieves 0% error rate for a 50-word window size.

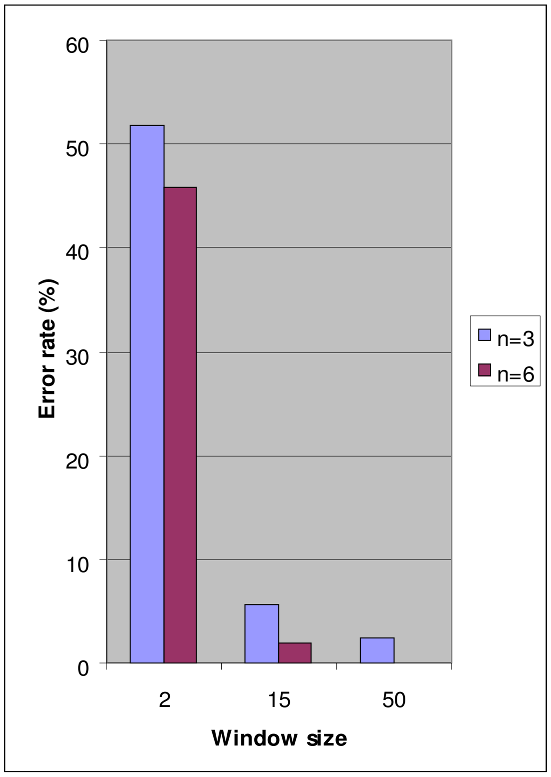

Figure 3: *Error rates of likelihood-based classifier for different values of n*

4.2. Cross-domain tests

For our cross-domain tests, the same training data as in Section 4.1 was employed, but now a test set was selected from the non-governmental documents. Typical results are shown in Table 2, which compares error rates within and across domains for the SVM and likelihood-based classifiers. We see that all classifiers are

Classifier	In-domain error rate	Cross-domain error rate
SVM, *n*=3	5.60%	20.70%
Likelihood-based, *n*=3	5.60%	29.00%
Likelihood-based, *n*=6	2.40%	14.40%

Figure 4: *Comparison of in-domain and cross-domain error rates for a window size of 15 words*

substantially less accurate when generalizing across domains. The SVM is somewhat more robust than the likelihood-based classifiers, but the best performance overall is again obtained with the likelihood-based, *n=6* classifier.

4.3. Size of training samples

Building the likelihood training models was less time consuming, so the relationship between error rate and the size of the training set was investigated for that classifier. Figures 5 to 7 show the error rates on the non-domain data. Enough text made it possible to train the window size of 2 words up to 10000 samples, but for the others it was only possible to use 3000 training samples. By using a larger training set the error rate can be reduced significantly. For *n*=6 the error rate at 3000 samples is 0.6% (and it seems that an increase in the number of training samples will decrease it even more), which is much lower than the equivalent number for the 200 sample experiments in Figure 4. An interesting observation is that for *n*=3, the error rates repeatedly reach a minumum, followed by an increase in error rate. This contradicts theoretical expectations, and is probably caused by peculiarities in the training and test data.

5. Conclusion

We have found that acceptable language-identification accuracy can be obtained with as few as 15 words of input text (in fact, even with 2 words somewhat useful results are obtained). In our tests, the SVM and likelihood-based classifiers perform with roughly similar accuracy when employed under the same circumstances, with the SVM generally somewhat preferable. However, the computational complexity of training the SVM with large numbers of features and samples is substantial – we were therefore not able to train the SVM with *n*=6 (which gave our best result for the likelihood classifier). The likelihood-based classifier may therefore be preferable in practice, because of its overall simplicity.

Our results clearly demonstrate that larger

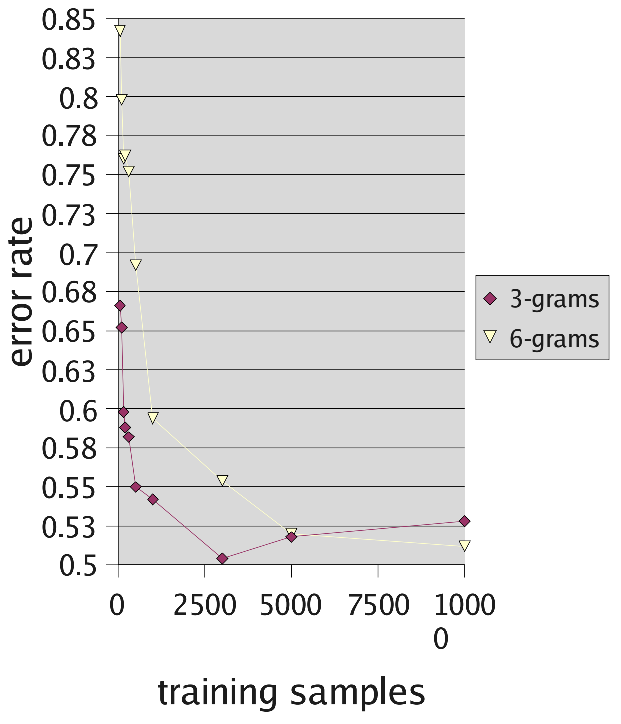

Figure 5: *Error rate for a window size of 2 words using various numbers of training samples*

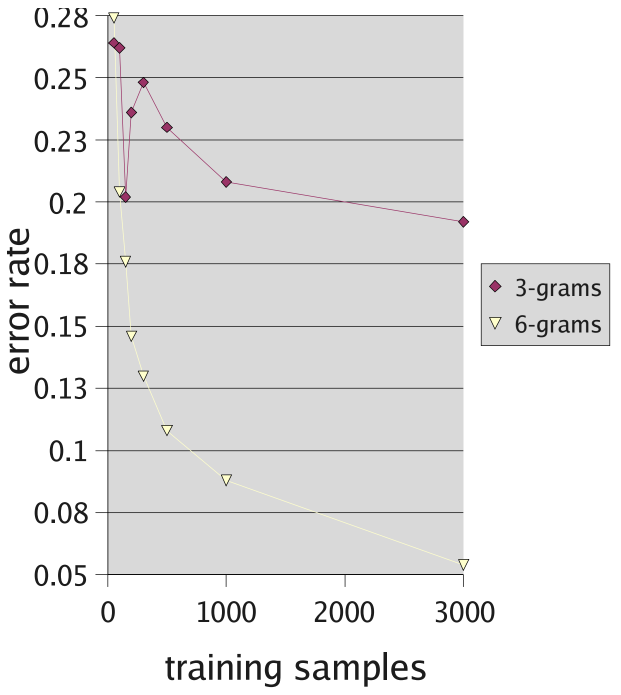

Figure 6: *Error rate for a window size of 15 words using various numbers of training samples*

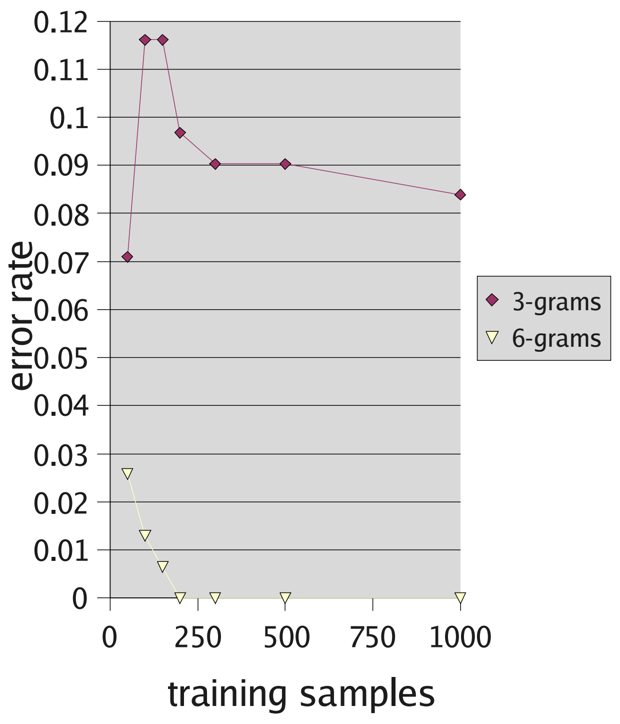

Figure 7: *Error rate for a window size of 50 words using various numbers of training samples*

values of *n* are preferable and in tests not reported here we have found *n*=6 to be the best value overall. The optimal value of this parameter will depend on the amount of training data available.

In [8] the average error rates for Danish, Dutch, English, French, Italian and, Spanish text was calculated by comparing the *n*-gram frequencies for different sizes of *n* (2-7). For a string of 100 characters an error rate of 1.03 % was obtained. This can roughly be compared with a window size of 15 words, where our error rate is 0.6%. In the only other research (to our knowledge) including the South African languages in a text-based language identification task, Combrink and Botha [9] reported substantially lower performance for the South African languages than for a set of European languages. Unfortunately, they do not report error rates, so it is not possible to compare results directly.

Our experiments with different training-set sizes suggest that the addition of more training data is the most straightforward route to improving the accuracy of our system. Further improvements are likely to result from a careful expansion of the symbol set (for example, by using the additional characters that occur in languages such as Sepedi and Setswana when they are available, and possibly by maintaining the distinction between lower- and upper-case characters). Finally, algorithmic improvements (e.g. by combining the likelihoods in more sophisticated ways) may also produce additional benefits.

6. References

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

[2] G. Botha and E. Barnards, "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.

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

[4] L. Zhai, M. Siu, X. Yang and H. Gish. (2006, June). Discriminatively Trained Language Models Using Support Vector Machines for Language Identification. [Online]. Available: http://www.ee.ust.hk/ eemsiu/Paper/svm-rev2.pdf

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

[6] I.H. Witten and E. Frank, "Data Mining: Practical Machine Learning Tools and Techniques", 2nd Edition, Morgan Kauffmann, Francisco, 2005.

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

[8] B. Ahmed, S. Cha and C. Tappert, "Language Identification from Text Using N-Gram Cumulative Frequency Addition", In. *Proceedings of CSIS Research Day*, Pace University, America, New York, pp. 12.1-12.8, 2004.

[9] 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.