Factors that affect the accuracy of text-based language identification

Gerrit R. Botha and Etienne Barnard

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

Abstract

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, using *n*-gram statistics as features for classification. For a fixed value of *n*, support vector machines generally outperform the other classifiers, but the simpler classifiers are able to handle larger values of *n*. This is found to be of overriding performance, and a Naïve Bayesian classifier is found to be the best choice of classifier overall.

For input strings of 100 characters or more 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.

1. Introduction

Although text-based language identification (LID) has been studied extensively, there is still not a general understanding of the factors that determine classification accuracy. In previous work [1] we identified three crucial factors, namely: the size of the textual fragment to identify, the amount and variety of training data and the classification algorithm employed. 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 [2]. 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, our results are the first comprehensive investigation into the accuracy achievable in text-based LID for the South African languages.

2. Background

2.1. 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. Statistics are gathered from characteristics such as letter sequences and conventional algorithms from pattern recognition are 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.

2.2. *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.

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* and the number of distinct "characters" in the ortography employed. 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.

2.3. Text-based language identification approaches

A variety of text-based LID approaches have appeared in the literature. A collection of simpler methods include: *n*-gram rank ordering [3], Naïve Bayesian (NB) [4], normalized dot-product [5], centroid-based [6] and relative entropy [7].

A range of modern pattern-recognition algorithms can also be applied. For example: support vector machines (SVMs) [8], Monte Carlo sampling [9], Markov models [10], decision trees [11], neural networks [12] and multiple linear regression [13].

In other approaches, methods such as: A third family of approaches leans more heavily on linguistic information - for example, by building language profiles from unique character string and most frequent words [14], using pure linguistic knowledge [15] and using a combination of linguistic knowledge and statistics [16].

3. Implementation

3.1. Classifiers and feature set

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 therefore 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 [10]. We therefore consider this classifier as a good baseline system for comparison with the other implemented classifiers.

The *n*-gram rank ordering method [3] is an approach which is referenced, used and compared in many studies. The method is mostly outperformed in comparison tests [6][17]. 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 [6], 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 [18] was used.

Based on their simplicity and excellent observed performance, we employed only *n*-gram statistics as 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.

As mentioned in Section 2.2, 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.

3.2. Datasets and sample sizes

Texts from various domains in all 11 South African languages were obtained from Professor D.J. Prinsloo of the University of Pretoria and using a web crawler [19]. The data included text from various sources (such as newspapers, periodicals, books, the Bible and government documents) and therefore, the corpus spans several domains. 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.

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 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.

Language	**Number of words**	**Average word length**	**15 charac-ters**	**100 charac-ters**	**300 charac-ters**
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 1: Word statistics on a 2M character file of each language. Average word lengths and number of words per character window are indicated.

3.3. Classifier implementations

3.3.1. Naïve Bayesian approach

For each language a vector of *n*-gram probabilities is computed by

$$\mathbf{l}_j = \frac{\mathbf{f}_j}{|\mathbf{f}_j|}, \tag{1}$$

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 is calculated in the logarithmic domain. In the next equation the log likelihood simplifies calculations by adding logarithmic probabilities and can be expressed as

$$P(L|D) = \sum_{i=1}^{n-\alpha+1} \ln l_j(c_i), \tag{2}$$

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.

3.3.2. Difference-in-frequency classifier

The same procedure as in the NB classifier Section 3.3.1 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

$$D_l = \sum_{i=1}^{n-\alpha+1} |l_j(c_i) - x(c_i)|, \tag{3}$$

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.

3.3.3. Support vector machine

The LIBSVM [18] library provides a full implementation of several SVMs. As discussed previously, the 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 [20] 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 validation data. These "reasonable" parameters were found in a preliminary investigation and 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, thus 55 classifiers were constructed) and each one trains from data of two different classes. 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.

4. Results

Our main results are shown in Figure 1, 2 and 3. These figures represent the error rates obtained with various classifiers as a function of the amount of training data for different window sizes.

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).

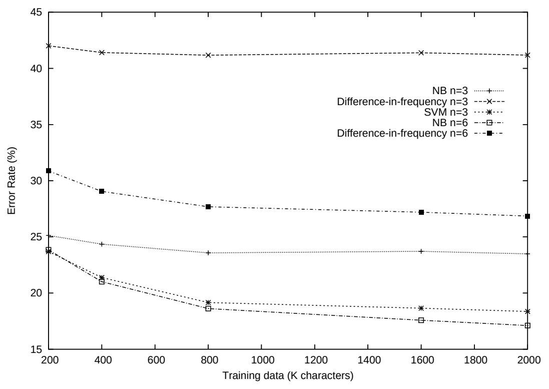

Figure 1: Classification results on 15 character window. Error rates are calculated by applying 10-fold cross-validation on various amounts of text.

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.

Figure 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 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.

From the results it is obvious that larger test windows

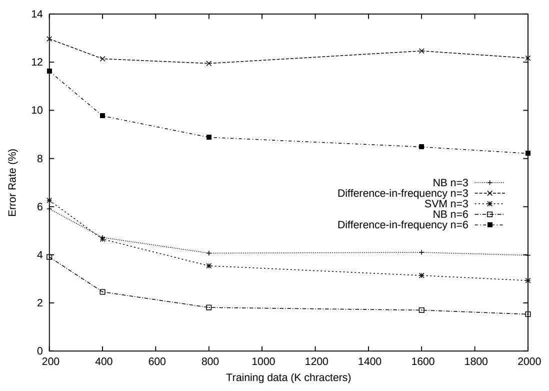

Figure 2: Classification results on 100 character window. Error rates are calculated by applying 10-fold cross-validation on various amounts of text.

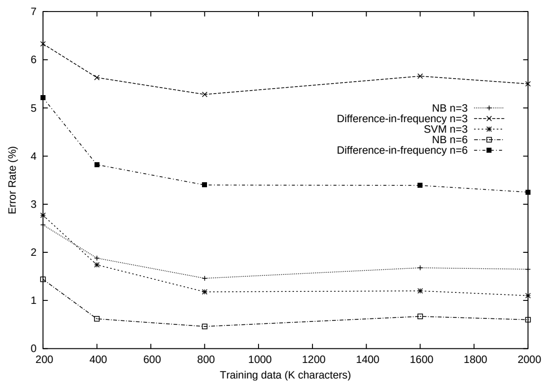

Figure 3: Classification results on 300 character window. Error rates are calculated by applying 10-fold cross-validation on various amounts of text.

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. 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 and 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 it is 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.

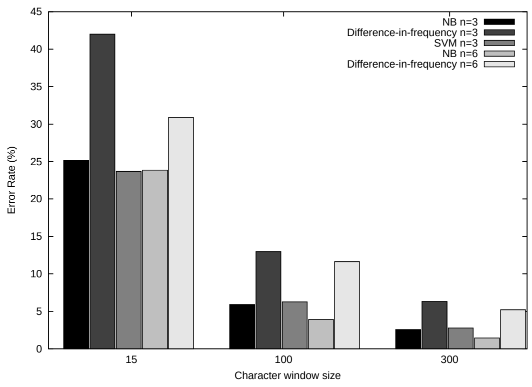

Figure 4: Error rates for different characters windows. This results were obtained using 10-fold cross-validation on 200K characters of text.

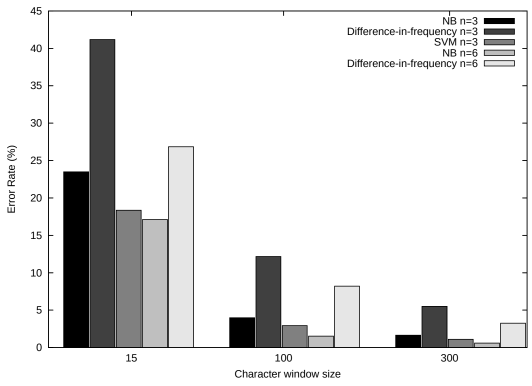

Figure 5: Error rates for different characters windows. This results were obtained using 10-fold cross-validation on 2M characters of text.

4.1. Confusion within language families

We found significantly higher error rates on the smallest (15-character) window. To understand the results better we analyzed the confusions between languages [21] and found (predictably) 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 2. (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.

Sotho family	Nguni family	TV	XT	AF	EN	
3837	423	160	208	72	157	Sotho Family
541	51096	237	375	107	317	Nguni Family
289	440	12158	270	15	46	TV
339	478	250	12028	22	78	XT
58	92	9	11	12876	232	AF
91	165	21	45	177	12608	EN

Table 2: 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 merged into families. TV=Tshivenda, XT=Xitsonga, AF=Afrikaans and EN=English.

Sotho family	Nguni family	TV	XT	AF	EN	Overall
2.59	2.99	8.02	8.44	3.03	3.81	4.88

Table 3: Error rates across language families, calculated from confusion matrices in Table 2.

5. Conclusion

All the classifiers showed improvement at larger training sets. From the smallest data set to the 1.6M dataset, a considerable reduction in error rate occurred in most cases. Error rates between the 1.6M and 2M data sets 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 substantially 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 proved to be inferior to the other two methods. The classifier also showed that the *6*-gram statistics improves accuracy over the smaller *n*-gram size.

Besides its excellent accuracy, 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.

6. References

[1] G. Botha, V. Zimu and E. Barnard, Text-based language identification for the South African languages, *Proceedings of the 17th Annual Symposium of the Pattern Recognition Association of South Africa*, Parys, pp. 46-52, November, 2006.

[2] 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.

[3] 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.

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

[5] M. Damashek, "Gauging similarity with n-grams: Language independent categorization of text," *Science*, Vol. 267, pp. 843848 , 1995.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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].

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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].

[19] 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.

[20] 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.

[21] G.R. Botha, "Text-based language identification for the South African languages," University of Pretoria, Masters dissertation, August, 2007.