Language identification of names with SVMs

Aditya Bhargava and Grzegorz Kondrak
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
{abhargava,kondrak}@cs.ualberta.ca

Abstract

The task of identifying the language of text or utterances has a number of applications in natural language processing. Language identification has traditionally been approached with character-level language models. However, the language model approach crucially depends on the length of the text in question. In this paper, we consider the problem of language identification of names. We show that an approach based on SVMs with n-gram counts as features performs much better than language models. We also experiment with applying the method to pre-process transliteration data for the training of separate models.

1 Introduction

The task of identifying the language of text or utterances has a number of applications in natural language processing. Font Llitjós and Black (2001) show that language identification can improve the accuracy of letter-to-phoneme conversion. Li et al. (2007) use language identification in a transliteration system to account for different semantic transliteration rules between languages when the target language is Chinese. Huang (2005) improves the accuracy of machine transliteration by clustering his training data according to the source language.

Language identification has traditionally been approached using character-level n-gram language models. In this paper, we propose the use of support vector machines (SVMs) for the language identification of very short texts such as proper nouns. We show that SVMs outperform language models on two different data sets consisting of personal names. Furthermore, we test the hypothesis that language identification can improve transliteration by pre-processing the source data and training separate models using a state-of-the-art transliteration system.

2 Previous work

N-gram approaches have proven very popular for language identification in general. Cavnar and Trenkle (1994) apply n-gram language models to general text categorization. They construct character-level language models using n-grams up to a certain maximum length from each class in their training corpora. To classify new text, they generate an n-gram frequency profile from the text and then assign it to the class having the most similar language model, which is determined by summing the differences in n-gram ranks. Given 14 languages, text of 300 characters or more, and retaining the 400 most common n-grams up to length 5, they achieve an overall accuracy of 99.8%. However, the accuracy of the n-gram approach strongly depends on the length of the texts. Kruengkrai et al. (2005) report that, on a language identification task of 17 languages with average text length 50 bytes, the accuracy drops to 90.2%. When SVMs were used for the same task, they achieved 99.7% accuracy.

Konstantopoulos (2007) looks particularly at the task of identifying the language of proper nouns. He focuses on a data set of soccer player names coming from 13 possible national languages. He finds that using general n-gram language models yields an average F_1 score of only 27%, but training the models specifically to these smaller data gives significantly better results: 50% average F_1 score for last names only, and 60% for full names.

On the other hand, Li et al. (2007) report some good results for single-name language identification using n-gram language models. For the task of separating single Chinese, English, and Japanese names, they achieve an overall accuracy of 94.8%. One reason that they do better is because of the smaller number of classes. We can further see that the languages in question are very dissimilar, making the problem easier; for example, the character “x” appears only in the list of Chinese names, and the bigram “kl” appears only in the list of English names.

3 Language identification with SVMs

Rather than using language models to determine the language of a name, we propose to count character n-gram occurrences in the given name, for n up to some maximum length, and use these counts as the features in an SVM. We choose SVMs because they can take a large number of features and learn to weigh them appropriately. When counting n-grams, we include space characters at the beginning and end of each word, so that prefixes and suffixes are counted appropriately. In addition to n-gram counts, we also include word length as a feature.

In our initial experiments, we tested several different kernels. The kernels that performed the best were the linear, sigmoid, and radial basis function (RBF) kernels. We tested various maximum n-gram lengths; Figure 1 shows the accuracy of the linear kernel as a function of maximum n-gram length. Polynomial kernels, a substring match–count string kernel, and a string kernel based on the edit distance all performed poorly in comparison. We also experimented with other modifications such as normalizing the feature vectors, and decreasing the weights of frequent n-gram counts to avoid larger counts dominating smaller counts. Since the effects were negligible, we exclude these results from this paper.

In our experiments, we used the LIBLINEAR (Fan et al., 2008) package for the linear kernel and the LIBSVM (Chang and Lin, 2001) package for the RBF and sigmoid kernels. We discarded any periods and parentheses, but kept apostrophes and hyphens, and we converted all letters to lower case. We removed very short names of length less than two. For all data sets, we held out 10% of the data

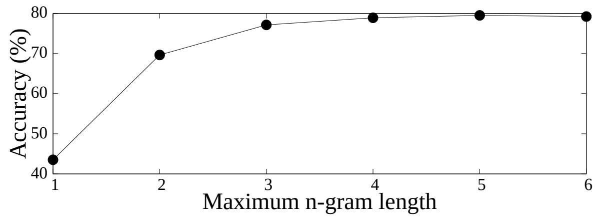

Figure 1: Cross-validation accuracy of the linear kernel on the Transfermarkt full names corpus.

as the test set. We then found optimal parameters for each kernel type using 10-fold cross-validation on the remaining training set. This yielded optimum maximum n-gram lengths of four for single names and five for full names. Using the optimal parameters, we constructed models from the entire training data and then tested the models on the held-out test set.

4 Intrinsic evaluation

We used two corpora to test our SVM-based approach: the Transfermarkt corpus of soccer player names, and the Chinese-English-Japanese (CEJ) corpus of first names and surnames. These corpora are described in further detail below.

4.1 Transfermarkt corpus

The Transfermarkt corpus (Konstantopoulos, 2007) consists of European soccer player names annotated with one of 13 possible national languages, with separate lists provided for last names and full names. Diacritics were removed in order to avoid trivializing the task. There are 14914 full names, with average length 14.8, and 12051 last names, with average length 7.8. It should be noted that these data are noisy; the fact that a player plays for a certain nation’s team does not necessarily indicate that his or her name is of that nation’s language. For example, *Dario Dakovic* was born in Bosnia but plays for the Austrian national team; his name is therefore annotated as German.

Table 1 shows our results on the Transfermarkt corpus. Because Konstantopoulos (2007) provides only F_1 scores, we used his scripts to generate new results using language models and calculate the accuracy instead, which allows us to be consistent with our tests on other data sets. Our results show that us-

Method	Last names	Full names
Language models	44.7	54.2
Linear SVM	**56.4**	**79.9**
RBF SVM	55.7	78.9
Sigmoid SVM	56.2	78.7

Table 1: Language identification accuracy on the Transfermarkt corpus. Language models have $n = 5$.

ing SVMs clearly outperforms using language models on the Transfermarkt corpus; in fact, SVMs yield better accuracy on last names than language models on full names. Differences between kernels are not statistically significant.

4.2 CEJ corpus

The CEJ corpus (Li et al., 2007) provides a combined list of first names and surnames, each classified as Chinese, English, or Japanese. There are a total of 97115 names with an average length of 7.6 characters. This corpus was used for the semantic transliteration of personal names into Chinese.

We found that the RBF and sigmoid kernels were very slow—presumably due to the large size of the corpus—so we tested only the linear kernel. Table 2 shows our results in comparison to those of language models reported in (Li et al., 2007); we reduce the error rate by over 50%.

Method	Ch.	Eng.	Jap.	All
Lang. model	96.4	89.9	96.5	94.8
Linear SVM	**99.0**	**94.8**	**97.6**	**97.6**

Table 2: Language identification accuracy on the CEJ corpus. Language models have $n = 4$.

5 Application to machine transliteration

Machine transliteration is one of the primary potential applications of language identification because the language of a word often determines its pronunciation. We therefore tested language identification to see if results could indeed be improved by using language identification as a pre-processing step.

5.1 Data

The English-Hindi corpus of names (Li et al., 2009; MSRI, 2009) contains a test set of 1000 names represented in both the Latin and Devanagari scripts. We manually classified these names as being of either Indian or non-Indian origin, occasionally resorting to web searches to help disambiguate them.[1] We discarded those names that fell into both categories (e.g. "Maya") as well as those that we could not confidently classify. In total, we discarded 95 of these names, and randomly selected 95 names from the training set that we could confidently classify to complete our corpus of 1000 names. Of the 1000 names, 546 were classified as being of Indian origin and the remaining 454 were classified as being of non-Indian origin; the names have an average length of 7.0 characters.

We trained our language identification approach on 900 names, with the remaining 100 names serving as the test set. The resulting accuracy was **80%** with the linear kernel, **84%** with the RBF kernel, and **83%** with the sigmoid kernel. In this case, the performance of the RBF kernel was found to be significantly better than that of the linear kernel according to the McNemar test with $p < 0.05$.

5.2 Experimental setup

We tested a simple method of combining language identification with transliteration. We use a language identification model to split the training, development, and test sets into disjoint classes. We train a transliteration model on each separate class, and then combine the results.

Our transliteration system was DirecTL (Jiampojamarn et al., 2009). We trained the language identification model over the entire set of 1000 tagged names using the parameters from above. Because these names comprised most of the test set and were now being used as the training set for the language identification model, we swapped various names between sets such that none of the words used for training the language identification model were in the final transliteration test set.

Using this language identification model, we split the data. After splitting, the "Indian" training, development, and testing sets had 5032, 575, and 483 words respectively while the "non-Indian" sets had 11081, 993, and 517 words respectively.

[1]Our tagged data are available online at `http://www.cs.ualberta.ca/~ab31/langid/`.

5.3 Results

Splitting the data and training two separate models yielded a combined top-1 accuracy of **46.0%**, as compared to **47.0%** achieved by a single transliteration model trained over the full data; this difference is not statistically significant. Somewhat counter-intuitively, using language identification as a pre-processing step for machine transliteration yields no improvement in performance for our particular data and transliteration system.

While it could be argued that our language identification accuracy of 84% is too low to be useful here, we believe that the principal reason for this performance decrease is the reduction in the amount of data available for the training of the separate models. We performed an experiment to confirm this hypothesis: we randomly split the full data into two sets, matching the sizes of the Indian and non-Indian sets. We then trained two separate models and combined the results; this yielded a top-1 accuracy of **41.5%**. The difference between this and the 46.0% result above is statistically significant with $p < 0.01$. From this we conclude that the reduction in data size was a significant factor in the previously described null result, and that language identification does provide useful information to the transliteration system. In addition, we believe that the transliteration system may implicitly leverage the language origin information. Whether a closer coupling of the two modules could produce an increase in accuracy remains an open question.

6 Conclusion

We have proposed a novel approach to the task of language identification of names. We have shown that applying SVMs with n-gram counts as features outperforms the predominant approach based on language models. We also tested language identification in one of its potential applications, machine transliteration, and found that a simple method of splitting the data by language yields no significant change in accuracy, although there is an improvement in comparison to a random split.

In the future, we plan to investigate other methods of incorporating language identification in machine transliteration. Options to explore include the use of language identification probabilities as features in the transliteration system (Li et al., 2007), as well as splitting the data into sets that are not necessarily disjoint, allowing separate transliteration models to learn from potentially useful common information.

Acknowledgements

We thank Sittichai Jiampojamarn for his assistance with the DIRECTL transliteration system, Shane Bergsma for his advice, and Stasinos Konstantopoulos for providing us with his scripts and data. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

References

W. B. Cavnar and J. M. Trenkle. 1994. N-gram-based text categorization. In *Proc. of the Third Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175.

C.-C. Chang and C.-J. Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

A. Font Llitjós and A. W. Black. 2001. Knowledge of language origin improves pronunciation accuracy of proper names. In *Proc. of Eurospeech*, pages 1919–1922.

F. Huang. 2005. Cluster-specific named entity transliteration. In *Proc. of HLT-EMNLP*, pages 435–442.

S. Jiampojamarn, A. Bhargava, Q. Dou, K. Dwyer, and G. Kondrak. 2009. DirecTL: a language independent approach to transliteration. In *Proc. of ACL-IJCNLP Named Entities Workshop*, pages 28–31.

S. Konstantopoulos. 2007. What's in a name? In *Proc. of RANLP Computational Phonology Workshop*.

C. Kruengkrai, P. Srichaivattana, V. Sornlertlamvanich, and H. Isahara. 2005. Language identification based on string kernels. In *Proc. of International Symposium on Communications and Information Technologies*.

H. Li, K. C. Sim, J.-S. Kuo, and M. Dong. 2007. Semantic transliteration of personal names. In *Proc. of ACL*, pages 120–127.

H. Li, A. Kumaran, V. Pervouchine, and M. Zhang. 2009. Report of NEWS 2009 machine transliteration shared task. In *Proc. of Named Entities Workshop: Shared Task on Transliteration*, pages 1–18.

MSRI, 2009. Microsoft Research India. `http://research.microsoft.com/india`.