LANGUAGE IDENTIFICATION OF INDIVIDUAL WORDS IN A MULTILINGUAL AUTOMATIC SPEECH RECOGNITION SYSTEM

Andrea Haţegan[1], *Bogdan Bârligă*[2], *Ioan Tăbuş*[1]

[1] Tampere University of Technology, FINLAND; [2] Nokia Devices, Tampere, FINLAND

ABSTRACT

This paper presents a new algorithm for identifying the language of words in a multilingual automatic speech recognition system. The new algorithm uses as input written words and it is composed of a method for language modelling and a method to select the language of a given word based on the available models. We also present two selection rules for the model's parameters. One of the rules uses a free parameter that controls the accuracy of the resulted model, as well as its size. On average, the classification accuracy of the new algorithm is above 70% for the first best and above 80% for the first two best.

Index Terms— Pattern classification, data compression

1. INTRODUCTION

The problem of language identification of words gained more focus with the rapid increase in deployment of multilingual automatic speech recognition (ASR) systems on mobile devices. Automatic language identification (LID) allows multilingual ASR systems to offer open vocabulary speech recognition, a must-have feature of modern ASR systems. Allowing the user to increase the vocabulary of the ASR system enables basic use-cases like speaker independent name dialing or voice commanding.

For this purpose the system needs to be able to construct the acoustic representation of the words in the vocabulary. In a typical architecture of multilingual ASR systems (like the one described in [1]) this is done in two stages: first the dictionary entry is translated from text to a phonetic representation and next the acoustic model used in recognition is built from the obtained phoneme sequence.

The LID algorithm is needed in the first step as the rules for text to phoneme translation are language dependent. Identifying the wrong language leads to using a wrong set of text-to-phoneme translation rules, resulting in an acoustic model that will not resemble the actual utterance and thus altering the recognition accuracy. In addition to the identification accuracy, as part of a multilingual system, a language identification algorithm has to deliver also on other aspects, like configuration flexibility and tight resource requirements.

The paper is organized as follows: in the next section we present the main approaches to the problem of language identification and point out their advantages and drawbacks; the third section presents the new algorithm; the fourth section presents the experimental results and in the last section some conclusions are drawn.

2. PREVIOUS WORK

Early methods for language identification were developed to identify the language of entire documents using *n*-grams and statistics of short words [2], [3], [4]. Discriminative *n*-grams are used in [5], where each word is decomposed into a plurality of non-overlapping *n*-grams that are subsequently checked for matching against regular expression like n-gram patterns learned from the language. The method also defines a voting scheme that combines the word-level decision in order to obtain a document-level decision. The learned n-gram patterns are constructed for each language with respect to the other competing languages such that they distinguish whether the given word belongs to the language, or not.

In [6] decision-trees are compared to an enhanced n-gram method, and in [7] the comparison includes also a neural network approach. In addition to the identification accuracy, the comparison refers also to the resource requirements (memory consumption) of the three methods. The results showed that the neural networks outperform the *n*-gram and the decision tree based methods, both in identification accuracy and in terms of memory requirements. An enhanced neural network based language identification method is introduced in [8], where the proposed neural network design allows for significant reductions in memory consumption with virtually no impact on the identification performance.

Despite the above mentioned advantages, the neural network based language identification in [8] has a practical drawback: there is one model for each configuration of supported languages. This means that a new model has to be built for each configuration, and that new language support cannot be added to the system in a flexible manner. Also in the case

This work was supported by the Academy of Finland (application number 213462, Finnish Programme for Centers of Excellence in Research 2006-2011) and the Graduate school in Electronics, Telecommunication and Automation (GETA). Data were provided by Nokia Devices, Tampere, FINLAND.

of removing the support of one or several languages from the configuration, the size of the model would not decrease accordingly. It can be stated that the discriminative abilities of the neural network come at the price of flexibility.

This flexibility price becomes important when the LID algorithm is part of a multilingual system that is deployed in a wide variety of configurations for the supported languages. This is the case for example of deploying ASR technology in mobile phones that ship to numerous locations and with many configurations.

We propose a new algorithm for language identification of individual words that has its root in the data compression field. The new algorithm dubbed *lidComp*, constructs a model for each language and then a given word is classified based on the average codelength resulted when it is encoded using the different models available. The identified language is the one who's model yields the shortest average codelength for the given word. We present two methods for selecting the parameters of each language model. The parameters are adjusted independently of the other languages available and one of the methods has a free parameter that can be used to adjust the model size and its accuracy. We tested the new algorithm for identifying the language of names as well as regular words, in a configuration of six European languages. On average, the classification accuracy is above 70% for the first best and above 80% for the first two best.

3. LIDCOMP - THE NEW ALGORITHM

3.1. Problem statement

The problem of language identification of individual words can be formalized as follows: given a word $W = w_1 \dots w_n$ and a set of models $\mathcal{M}_i$ for each language, choose the language i^* for the given word, the one that minimizes the average codelength $L(W; \mathcal{M}_i)$ when encoding it with $\mathcal{M}_{i^*}$, i.e.:

$$i^* = \arg\min_i L(W; \mathcal{M}_i). \tag{1}$$

The language identification problem formalized like in (1) is in line with the minimum description length (MDL) principle [9]: select the hypothesis that minimize the number of bits needed to write the observed sequence. However, the accuracy of the proposed language selection rule depends on the descriptive power of the models used to represent each language. Once the models are set, the average codelength of a given word can be obtained using the arithmetic coding [10].

The models used by *lidComp* are tree machines (TM) introduced for data compression purposes in [11],[12] and further developed in [13]. In a data compression framework like in [11] and [12], a TM is adaptively built as the given data is encoded. The final TM is not used anymore after the encoding process is done. In [13] a TM is built from a training set and then it is used to encode small sequences, e.g. SMS, that have the same statistical properties like the training data, but no decision making is done.

In [11],[12], it has been shown that TM are optimal in the sense that they are able to compress long strings with a near optimum per symbol length without prior knowledge of the source generating the data. This means that after seeing enough data, the TM model incorporates the rules of the machinery that generated the data. Such property makes TMs strong candidates for language modelling and classification tasks. In the next subsection we describe the encoding process of a word using a TM and how the TM is built from a training set.

3.2. Coding and modelling

In a TM one node represents one context and each node/context keeps the probability distribution of the symbols in the alphabet seen in that context. Thus, the average codelength in (1) of a word W given a TM $\mathcal{M}_i$ is computed as follows:

$$L(W; \mathcal{M}_i) = -\sum_{j=1}^{n} \log_2 P(w_j | c(w_j); \mathcal{M}_i), \tag{2}$$

where $c(w_j)$ denotes the context for the current symbol w_j. The context for the symbol w_j is found by climbing the tree starting from root and then following the branch w_{j-1}, at the arrived node following the branch w_{j-2}, and so on, until a leaf or the first symbol w_1 of the word is reached. It is possible to stop also in an internal node in the tree, if w_{j-k} is not a branch of the node $w_{j-1} \dots w_{j-k+1}$.

A TM is obtained by pruning a maximal tree machine (MTM) that is collected on the training set. The pruning process is needed because the resulted MTM is usually over fitted to the training set and our goal is to find a model that generalizes for data that were not used for training. Next, we present the MTM building process and how a TM is obtained from it for one language. The process is repeated for each given language.

The MTM is built from a large collection of words in two steps: context assignment and symbol occurrence collection. Figure 1 illustrates the process of context assignment: first, each word is padded with the start/stop symbol and then for each symbol in all words we keep its prefix; the prefixes are sorted in lexicographical order with priority on the right; finally, the contexts that form the MTM are obtained by retaining from each prefix the rightmost symbols such that the context becomes deterministic, i.e. in each context only one symbol is seen. One context might not be deterministic if the entire prefix has to be taken as the context, e.g. "$\star\underline{\mathrm{x}}$". Once the contexts are available, the MTM is obtained as follows: start with the empty tree, i.e. the root and all the symbol counts equal to zero. For each pair (context, symbol) add the context to the tree by incrementing the count of the symbol in all the nodes on the path from the root to the context. The resulted

MTM contains all the contextual information available in the training data.

If the resulted MTM would be used to encode words that are not in the training data, the *zero-frequency* problem will occur, i.e. it is possible that in some nodes the probability of several symbols to be zero. We tackle this problem by introducing the escape probability like in [13] and in this way we assign a probability different than zero to each symbol in the alphabet, at all nodes.

By the context selection rule we select the deepest node in the tree for each symbol that has to be encoded. Although the longer contexts (deep nodes) contain more relevant statistics than the shorter ones, the information they provide is less reliable, because usually there is a limited amount of training data and the statistics accumulate slower. Thus, we would like to avoid long contexts that are not reliable and fitted to the training set by pruning the MTM. The resulted pruned tree is the language model in the form of a TM.

We experimented with two pruning strategies: *mdlPrune* and *freePrune*. The first one starts in the root and splits nodes until a given condition is true, while the other pruning strategy starts at the leaves and prunes nodes until another condition is true. The second pruning rule uses a free parameter that controls how much the MTM is pruned.

Denote by $n_{j|s}$ the frequency of the jth symbol in the alphabet at node s and by $\hat{P}_{j|s}$ its probability that was adjusted to include also the escape probability. Also, denote by $n_s = \sum_j n_{j|s}$ the total number of symbols seen at node s and by $N_{\mathcal{A}} = |\mathcal{A}|$, the alphabet's cardinality. The children of node s are denoted by $\mathcal{C}(s)$.

3.2.1. *mdlPrune*

For each node s in the resulted MTM compute $L_s = -\sum_j n_{j|s} \log_2 \hat{P}_{j|s} + \frac{N_{\mathcal{A}}}{2} \log_2 n_s$. This is the two part code description length [9] of the symbols seen at node s. Starting in the root, for each node s prune all its children if $L_s < \sum_{r \in \mathcal{C}(s)} L_r$. According to the MDL principle, by this pruning rule, we retain in the model the nodes that best describe the data, by penalizing the nodes that are too expensive and thus over fitted to the training set.

3.2.2. *freePrune*

For each internal node r in the final MTM set $L(r) = 0$ and for each leaf s compute $L(s) = -\sum_j n_{j|s} \log_2 \hat{P}_{j|s}$. The idea is to compute for each internal node, the shortest codelength of the symbols that appear at all its descendants. Thus, starting at the father nodes of the leaves, for a given internal node r and each of its children s, compute $I_r(s) = -\sum_j n_{j|s} \log_2 \hat{P}_{j|r}$. $I_r(s)$ represents the codelength of the symbols seen at s if they would be encoded with the distribution of the symbols seen at the father node r. If $I_r(s) \leq (1+p)L(s)$, then the codelength computed at the father node is not that worse, and thus we can prune s and update $L(r) = L(r) + I_r(s)$. If $I_r(s) > (1+p)L(s)$, the parent node assignment is not worthy and thus we keep s and update $L(r) = L(r) + L(s)$. The free parameter p controls the accuracy of the resulted TM, as well as its size.

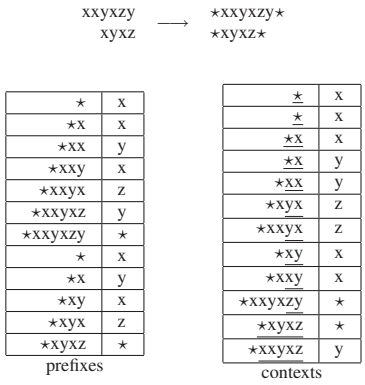

Fig. 1. Context assignment in the MTM building process

We have described by now all the ingredients of the new algorithm *lidComp*: how to obtain a model for a given language from a training set in the form of a TM and how to compute the codelength of a word based on a given model. The predicted language for the given word is the one that yields the shortest description length.

4. RESULTS

The performance of the new algorithm *lidComp* was evaluated for a configuration of six European languages: French (fre), German (ger), Italian (ita), Portuguese (por), Spanish (spa) and English (uk).

Table 1 presents the data that were used for our experiment. We tested the *lidComp*'s accuracy on *names* and *words*. The "calibrating" set is used to set the free parameter p for the *freePrune* pruning method, as follows: for each language prune the MTM using different values for p and then, the TM that best encodes the calibration set is selected as the language model. All the resulted models are built independent of the other languages in the configuration. For each language, the three data sets are distinct.

Names	fre	ger	ita	por	spa	uk
training	4635	4534	3988	3666	4654	3186
calibrating	1825	1767	1572	1424	1837	1238
testing	2809	2766	2416	2242	2817	1948
Words	**fre**	**ger**	**ita**	**por**	**spa**	**uk**
training	9073	8783	7493	4911	7566	5897
calibrating	3577	3466	2925	1927	2964	2335
testing	5495	5316	4567	2983	4602	3562

Table 1. European languages - the number of names/words used for training, calibrating, and testing.

The accuracy of the new algorithm is measured by the

percentage of correct classified words for a given language, i.e. how many times the true language yielded the *first best* score (part (a) in Tables 2,3). We also report the results when the true language has one of the *first two best* scores (part (b) in Tables 2,3). This is meaningful in the context of ASR systems since the language identification is prone to making mistakes. Thus, the ASR system can build more (usually two) acoustic models for the same vocabulary entry to increase the chances of having the correct acoustic representation.

We have used two training/testing scenarios: *distinct* (Table 2) where the training and test set for each language are distinct like in Table 1 and *common*(Table 3) where the training set is like in Table 1, but the test set contains all the available data: training, calibrating, and testing sets.

(a)

Names	fre(%)	ger(%)	ita(%)	por(%)	spa(%)	uk(%)	avg(%)
freePrune	73.87	**88.29**	**74.79**	**49.60**	72.70	**72.23**	**71.91**
mdlPrune	**80.88**	87.42	73.14	44.47	**75.36**	59.96	70.21
Words	fre(%)	ger(%)	ita(%)	por(%)	spa(%)	uk(%)	avg(%)
freePrune	**74.25**	**87.79**	82.66	**70.23**	**69.56**	**76.61**	**76.85**
mdlPrune	71.96	86.49	**85.29**	55.75	65.60	71.06	72.69

(b)

Names	fre(%)	ger(%)	ita(%)	por(%)	spa(%)	uk(%)	avg(%)
freePrune	85.90	**93.89**	**89.82**	76.81	87.22	**86.19**	86.64
mdlPrune	**90.74**	93.67	85.80	**80.78**	**89.28**	79.67	**86.66**
Words	fre(%)	ger(%)	ita(%)	por(%)	spa(%)	uk(%)	avg(%)
freePrune	86.42	**93.60**	92.25	**87.60**	87.40	**89.81**	**89.51**
mdlPrune	**87.63**	92.85	92.25	84.55	**87.90**	88.57	88.96

Table 2. Results on *distinct* sets: first(a) and first two(b) best.

The results on *distinct* sets are presented in Table 2. On average, the two pruning rules lead to similar performance on classifying *names* as well as *words*. In [8], the accuracy on *words* of the proposed neural network based LID, was tested for 25 languages. The classification accuracy varied between 57.36%-71.01% for the *first best*, depending on the configuration used. The accuracy for the *first two* best varied between 75.03%-82.99%. Unfortunately, we couldn't test *lidComp* algorithm on the data set used in [8] because the data were not available, but the percentages we obtain on our data set are superior with more than 5% when using *freePrune* (see *Words* in Table 2).

(a)

Names	fre(%)	ger(%)	ita(%)	por(%)	spa(%)	uk(%)	avg(%)
freePrune	**84.14**	**92.48**	**80.27**	**64.55**	**83.56**	**81.37**	**81.06**
mdlPrune	81.34	88.52	73.17	45.46	76.79	62.19	71.25
Words	fre(%)	ger(%)	ita(%)	por(%)	spa(%)	uk(%)	avg(%)
freePrune	**83.52**	**92.01**	**89.81**	**81.06**	**81.43**	**85.16**	**85.50**
mdlPrune	71.89	87.16	85.69	56.25	67.03	72.78	73.47

(b)

Names	fre(%)	ger(%)	ita(%)	por(%)	spa(%)	uk(%)	avg(%)
freePrune	**92.93**	**96.53**	**92.44**	**86.96**	**93.03**	**91.82**	**92.20**
mdlPrune	91.09	94.51	86.11	81.60	89.68	80.52	87.25
Words	fre(%)	ger(%)	ita(%)	por(%)	spa(%)	uk(%)	avg(%)
freePrune	**92.19**	**95.98**	**95.85**	**92.87**	**93.24**	**94.11**	**94.04**
mdlPrune	87.61	93.07	92.96	84.60	88.67	89.65	89.43

Table 3. Results on *common* sets:first(a) and first two(b) best.

Table 3 presents the classification accuracy, when the test set is composed half of the training data and half of new data. This experiment was done because in practice it can be expected that some vocabulary entries were used also in training. Comparing Table 2 and Table 3, it can be seen that the accuracy of *lidComp* when using *mdlPrune* is only marginally improved, while the accuracy when using *freePrune* is improved with more than 8% for the *first best* and with more than 5% for the *first two best*.

The experiments presented here were carried out using Matlab and the models are kept as ".mat" files. When pruning with *mdlPrune*, the size for all six models is 14.7kB for *names* and 25.6kB for *words*. When pruning with *freePrune*, the size for all six models is 113kB for *names* and 347kB for *words*. The size of the models can be reduced by a more appropriate representation than the one used by Matlab to encode ".mat" files, as well as by using different p values for the *freePrune* pruning method.

5. CONCLUSIONS

In this paper we addressed the language identification of words in the context of multilingual ASR systems. We presented a new algorithm *lidComp* that offers configuration flexibility as well as high classification accuracy. The results show that tree machines, successfully used in the data compression field, can be used also in classification tasks.

6. REFERENCES

[1] J. Iso-Sipila, M. Moberg, and O. Viikki, "Multi-lingual speaker-independent voice user interface for mobile devices," in *ICASSP 2006*. IEEE, 2006, vol. I, pp. 1081–1084.

[2] G. Greffenstette, "Comparing two language identification schemes," in *3rd International Conference on Statistical Analysis of Textual Data*, 1995.

[3] J. Prager, "Linguini: language ldentification for multilingual documents," in *32nd Hawaii International Conference on System Sciences*, 1999, pp. 1–11.

[4] J. C. Schmitt, "Trigram-based method of language identification," *U.S. Patent number: 5062143*, 1991.

[5] M. Kantrowitz, "Method for identifying the language of individual words," *U.S. Patent number: 6292772*, 1998.

[6] J. Hakkinen and J. Tian, "*n*-gram and decision tree-based language identification for written words," in *IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2001.

[7] J. Tian, J. Hakkinen, S. Riis, and K. Jensen, "On text-based language identification for multilingual speech recognition systems," in *Proceedings of 7th International Conference on Spoken Language Processing*, 2002, pp. 501–504.

[8] J. Tian and J. Suontausta, "Scalable neural network based language identification from written text," in *ICASSP 2003*. IEEE, 2003, vol. I, pp. 48–51.

[9] J. Rissanen, "Modeling by the shortest data description," *Automatica*, vol. 14, pp. 465–471, September 1978.

[10] J. Rissanen, "Generalized kraft inequality and arithmetic coding," *IBM J. Res. Dev*, vol. 20, pp. 198–203, May 1976.

[11] J. Rissanen, "A universal data compression system," *IEEE Trans. on Information Theory*, vol. IT-29, pp. 656–664, September 1983.

[12] M.J. Weinberger, J. Rissanen, and M. Feder, "A universal finite memory source," *IEEE Trans. on Information Theory*, vol. IT-41, pp. 643–652, May 1995.

[13] J. Rissanen, "A lossless data compression system," *U.S. Patent number: 7028042*, 2006.