ON TEXT-BASED LANGUAGE IDENTIFICATION FOR MULTILINGUAL SPEECH RECOGNITION SYSTEMS

Jilei Tian\textsuperscript{1}, Juha Häkkinen\textsuperscript{2}, Søren Riis\textsuperscript{3}, Kåre Jean Jensen\textsuperscript{4}

\textsuperscript{1}Nokia Research Center, Speech and Audio Systems Laboratory, Tampere, Finland
\textsuperscript{2}Nokia Mobile Phones, Tampere, Finland
\textsuperscript{3}Oticon A/S, Copenhagen, Denmark
\textsuperscript{4}Nokia Mobile Phones, Copenhagen, Denmark
{jilei.tian, juha.m.hakkinen, kaare.jensen}@nokia.com, skr@oticon.dk

ABSTRACT

The demand for multilingual speech recognition systems is growing rapidly. Automatic language identification is an integral part of multilingual systems that use dynamic vocabularies. Most state-of-the-art automatic language identification approaches identify the language based on probabilities of phoneme sequences extracted from the acoustic signal. Such methods can, however, not be applied to language identification based on text alone. This paper compares three text-based language identification methods aimed particularly at very short segments of text as encountered in, e.g., name dialling or command word control applications. The first method is based on artificial neural networks, the second on decision trees and the third on \textit{n}-gram letter statistics. We describe how to apply neural networks to the LID task. After that, we briefly review both the decision tree and the \textit{n}-gram based approaches is evaluated in a number of text based tests, and the performance and complexity.

1. INTRODUCTION

In [10], we proposed a low complexity architecture for multilingual speech recognition in which the user is allowed to modify the active vocabulary. This system is useful for, e.g., small vocabulary speech recognition applications in portable terminals targeted for world-wide markets.

The multilingual speech recognition engine consists of three key modules: automatic language identification (LID) from text, on-line language-specific text-to-phoneme modeling (TTP), and multilingual acoustic modeling modules as shown in Figure 1. When the user adds a new word to the active vocabulary, a language tag is first assigned to this word by the LID module. Based on this language tag, the appropriate language-specific TTP model is applied to get the phoneme sequence associated with the written form of the vocabulary item. Finally, the recognition model for each vocabulary entry is constructed by concatenating the multilingual acoustic models according to the phonetic transcription.

Automatic LID can be divided into two classes: spoken- and text-based LID, i.e., language identification from speech or written text. Most speech based LID methods use a phonotactic approach, where sequences of phonemes are extracted from the speech signal using more or less standard speech recognition methods, such as HMMs. These phoneme sequences are then rescored by language-specific statistical models, such as \textit{n}-grams. By assuming that the language identity can be discriminated well by the characteristics of the phoneme sequences patterns, the rescoring will yield the highest score for the correct language. Language identification from text is commonly solved by gathering language specific \textit{n}-gram statistics for letters in the context of other letters [2][5][7]. Based on these language specific statistics, a score for each language can be computed for a chunk of text.

While the \textit{n}-gram based approach works quite well for fairly large amounts of input text (e.g., 10 words or more), it tends to break down for very short segments of text. This is especially true if the \textit{n}-grams are collected for common words and applied to identifying the language tag of a proper name. Proper names have very atypical grapheme statistics compared to common words as they often origin from different languages.

For short segments of text, other methods for LID might be more suitable. Decision trees and \textit{n}-grams were compared in text-based language identification in [3]. In this paper, we first describe how to apply neural networks to the LID task. After that, we briefly review both the decision tree and the \textit{n}-gram based LID methods. The performance of the proposed approaches is evaluated in a number of text based tests, and the
overall scheme is verified in a speech recognition test. As our focus is on speech recognition applications for mobile devices with limited memory and computational resources, both the performance and the complexity of the LID method play equally important roles.

2. LANGUAGE IDENTIFICATION METHODS

2.1. Neural network based LID

A simple neural network model which has successfully been applied to text-to-phoneme mapping is the multi-layer perceptron (MLP) [4]. As TTP and LID are similar tasks, this architecture may also be well suited for LID. The MLP is composed of layers of units (neurons) arranged so that information flows from the input layer to the output layer of the network. The basic neural network-based LID model is a standard two-layer MLP, as seen in Figure 2.

![Figure 2. Two-layer neural network architecture.](image)

Letters are presented to the MLP network one at a time in a sequential manner. The network gives estimates of language posterior probabilities for each presented letter. In order to take the grapheme context into account, a number of letters on each side of the letter in question can also be used as input to the network. Thus, a window of letters is presented to the neural network as input. Figure 2 shows a typical MLP with a context size of four letters \(l_{-4}...l_4\) centered at letter \(l_0\). The centermost letter \(l_0\) is the letter that corresponds to the output of the network. Therefore the output of the MLP is the estimated language probability for the centermost letter \(l_0\) in the given context \(l_{-4}...l_4\). A graphemic null is defined in the character set as the same words are sometimes used across various languages. Although written in the same way, such words are often pronounced very differently from language to language.

![Table 1. Orthogonal letter coding scheme.](image)

In Equation (1), \(y_i\) and \(P_i\) denote the \(i^{th}\) output value before and after softmax normalization. \(C\) is the number of units in the output layer, representing the number of classes, or target languages in our case. It has been shown in [1] that a neural network with softmax normalization will approximate class posterior probabilities when trained for one-out-of-\(N\) classification and when the network is sufficiently complex and trained to a global minimum. Since the neural network input units are continuous valued, the letters in the input window need to be transformed to some numeric quantity. An example of an orthogonal code-book representing an alphabet used for language identification is shown in Table 1. The last row in the table is the code for the graphemic null. The orthogonal code has an equal size to the number of letters in the alphabet. An important property of the orthogonal coding scheme is that it does not introduce any correlation between different letters.

The LID neural network is trained by the standard back-propagation (BP) algorithm augmented by a momentum term. Each letter with context and the corresponding language tag of the word make up one training example. Weights are updated in a stochastic on-line fashion, i.e., the weights are updated after presentation of each training example picked at random from the training set which has been shuffled. Before testing the models, all parameters are rounded off to eight bits as this was found sufficient for representing model parameters without a significant loss in accuracy. The number of parameters in the models therefore equals the required memory for storage in bytes.

The outputs of the LID neural network approximate the language posterior probabilities corresponding to the centermost letter. The language tag of a word is obtained by combining the network outputs for each individual letter in the word. Given a word with orthographic representation \(l_1, l_2, \ldots, l_N\), the language \(\text{lang}\) is given by

\[
\text{lang} = \arg\max_{\text{lang}} \left\{ \prod_{i=1}^{N} P_k(l_i^{+w}) \right\}^{1/N},
\]

where \(P_k(l_i^{+w})\) is the network output corresponding to language \(k\) given input letters \(l_i^{+w} = l_{i-w}, \ldots, l_i, \ldots, l_{i+w}\), and variable \(w\) denotes the letter window context size, respectively.

Obviously, language identities are not always unambiguous as the same words are sometimes used across various languages. Although written in the same way, such words are often pronounced very differently from language to language.
Therefore, it is often beneficial to include not only the transcription corresponding to the single best language, but transcription corresponding to the $N$ highest scoring languages. Of course, a list of the $N$-best languages can be generated by ranking the results from Equation (2).

2.2. Decision tree based LID

Decision trees have successfully been applied to tasks like text-to-phoneme mapping as in, e.g., [8] and also language identification [3]. Similarly to the neural network approach, decision trees are used for determining the language tag for each of the letters in a word. Contrary to the neural network approach, however, there is one decision tree for each of the different characters in the alphabet. The language tag for a letter is obtained by “asking” a series of questions about the context of the letter in question as defined by the corresponding decision tree. A decision tree is composed of a root node, internal nodes and leaves. In the trees used here, the context is defined by the neighboring letters. Each node contains information about the attribute and language identity.

In the decoding phase, a language-tag sequence is generated by going through the word letter by letter from left to right. The decision tree corresponding to the current letter is climbed based on the context information until a leaf is reached. The language tag that corresponds to the current letter is read from the leaf. Then the process moves on to the next letter and the language tag for this letter is found in a similar way. The language tag of the word is defined by the most frequent language tag for the letters of the word.

When a decision tree is trained for a given letter, all the training cases for the letter are considered. A training case for the letter is composed of the letter context and the corresponding language tag of the word. During training, the decision tree is grown and the nodes of the decision tree are split into child nodes according to an information theoretic optimization criterion [6]. Details about decision tree training can be found in [3][8].

2.3. $N$-gram based LID

The $n$-gram method uses letter $n$-grams, representing the frequency of occurrence of various $n$-letter combinations in a particular language. The letter $n$-grams for language lang$_k$, $P(l_i | t_{i−n+1}^{i−1}, lang_k)$, are estimated using text data from that language only.

Given a word $l_1, l_2, ..., l_p$, the language lang is given by Equation (3).

$$lang = \arg\max_{lang} \prod_{i=1}^{p} P(l_i | t_{i−n+1}^{i−1}, lang_i),$$

where given letter sequence is denoted by $t_{i,n}^{i−1} = l_{i−n+1}, ..., l_{i−1}$.

3. EXPERIMENTAL EVALUATION

3.1. Setup

The decision tree, $n$-gram and neural network based LID models are evaluated on four different languages: English, German, Spanish and Finnish. The text data is extracted from a balanced training set constructed from the following resources: CMUdict (US English), SpeechDat-Car-Finnish (Finnish), LDC-Callhome German and Spanish. The training set contains 60,750 words corresponding to approximately 15,000 words for each language. An independent test set of 124,918 words is constructed from the remaining items in the above mentioned resources. As the resources contain generic words and only a few proper names, they are called generic databases (GDB).

As mentioned above, proper names have a very atypical grapheme statistics compared to common words. Thus, LID for names is a much harder problem. Sometimes the same name is used in several different languages. For evaluating the performance on names, an in-house name database (NDB) consisting of more than 10,000 name entries in the four languages is used.

Both the decision tree and neural network methods use a context of 4 letters to the left and right of the centermost letter. The neural network uses 30 hidden units and four output units. The number of inputs is 369 corresponding to a 9-letter input window and 41 different characters (including the null character). With 8-bit precision for the network weights this corresponds to a memory requirement of about 11 kB. For comparison the size of the decision tree after training is 50 kB. The trigram takes up more than 100 kB. It should be noted that since pruning or compression is not applied on any of the methods the memory consumption results are only tentative.

3.2. Text-based language identification evaluation

Table 2 shows the language identification rates obtained by using neural network based LID (NN-LID), decision tree based LID (DT-LID) and $n$-gram based LID (NG-LID), respectively. The training is conducted on the GDB training set. Clearly the DT-LID memorizes the training set better than the NN-LID and NG-LID (91.9% vs. 87.2% and 80.4%). However, the NN-LID is far better than DT-LID and NG-LID (86.6% vs. 75.7% and 78.7%) when it comes to generalization performance on the independent test set. This is also reflected by the better performance on the names data set NDB, where the NN-LID is clearly superior to the DT-LID and NG-LID.

<table>
<thead>
<tr>
<th>Train on GDB train set</th>
<th>NN-LID</th>
<th>DT-LID</th>
<th>NG-LID</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDB train set</td>
<td>87.2%</td>
<td>91.9%</td>
<td>80.4%</td>
</tr>
<tr>
<td>GDB test set</td>
<td>86.6%</td>
<td>75.7%</td>
<td>78.7%</td>
</tr>
<tr>
<td>NDB</td>
<td>60.3%</td>
<td>52.2%</td>
<td>56.4%</td>
</tr>
</tbody>
</table>

Table 2. Language identification rates by using NN-LID, DT-LID and NG-LID, trained on GDB training.

Table 3 shows that both NN-LID and NG-LID performances increase quite dramatically when it is only required that the correct language tag is among the 2-best language tags provided by the LID. Naturally, if all 4 languages are included the performance will increase to 100%. For a limited number of supported languages, it might be possible to include transcriptions for all languages. However, as the number of supported languages increases the computational complexity and memory consumption may very well become prohibitive. Theoretically, $N$-best concept can be introduced to DT-LID as well, but it does not work in practice.
since one language quite often dominates the generated language tag sequence, actually there is no other alternative.

<table>
<thead>
<tr>
<th></th>
<th>NN-LID 2-best</th>
<th>NG-LID 2-best</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train on GDB train set</td>
<td>97.0%</td>
<td>95.0%</td>
</tr>
<tr>
<td>GDB train set</td>
<td>96.7%</td>
<td>94.0%</td>
</tr>
<tr>
<td>GDB test set</td>
<td>83.7%</td>
<td>81.3%</td>
</tr>
</tbody>
</table>

Table 3. 2-best results using NN-LID and NG-LID, trained on the GDB training set.

The experiments show that NN-LID outperforms DT-LID and NG-LID in terms of memory consumption and generalization capability. Concerning to the learning capability, NN-LID still outperforms DT-LID and NG-LID when $n$-best results enabled ($n=2$ in our tests).

### 3.3. Evaluation on speech recognition

To evaluate the effect of LID errors, the NN-LID model is evaluated as part of the multilingual recognizer shown in Figure 1. Details about this recognizer can be found in [10]. The evaluation is done for clean speech on a vocabulary composed of names. The majority of the names are full names (first name and surname), but also a number of first names are included to simulate a typical real-world usage pattern.

![Recognition rates with and without LID](image)

*Figure 3. The recognition results tested in clean speech database using NN-LID for four languages.*

Since LID is not always unambiguous as the same names are used across various languages, and the automatic process makes occasionally identification errors, we have proposed a bi-lingual scheme in [9], i.e., two languages are supported for each vocabulary item. First, the language corresponding to the user's native language is defined as the default language. The second language is generated based on the output of the NN-LID denoted by 1-best LID. Experiments are carried out to determine the effect of NN-LID on the recognition accuracy. The language identity of each vocabulary item is specified either manually by human expert, or using NN-LID. In the case of NN-LID, both 1-best, 2-best LID and bi-lingual schemes were investigated. As it is seen in the Figure 3, the use of 1-best LID degrades the recognition performance compared to the baseline system. By providing two language identity decisions for each vocabulary item, the recognition performance is significantly improved, and the bi-lingual scheme is even marginally better than the baseline system which uses manual LID.

### 4. CONCLUSIONS

The identification of the language of speech recognition vocabulary items directly from written text is an important problem in multilingual speech recognition applications. In this paper we have compared decision tree, neural network and $n$-gram based language identification methods for short segments of text. Even though the decision tree based method outperforms the neural networks in terms of memorizing the training material, our results show that the neural network provides clearly better overall performance and smaller model size. Especially the generalization performance of neural networks is quite superior. The performance of the neural network based language identification scheme is also verified in a multilingual speech recognition task.

### 5. REFERENCES