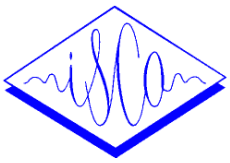

6th International Conference on Spoken Language Processing (ICSLP 2000)
Beijing, China
October 16-20, 2000

DECISION TREE BASED TEXT-TO-PHONEME MAPPING FOR SPEECH RECOGNITION

Janne Suontausta and Juha Häkkinen

Speech and Audio Systems Laboratory, Nokia Research Center, Tampere, Finland
Email: juha.m.hakkinen@nokia.com

ABSTRACT

In many embedded speech recognition systems, the phonetic transcriptions of the vocabulary items, i.e., the lexicons, cannot be stored to the device beforehand. A text-to-phoneme mapping functionality is hence needed to create the transcriptions from plain text. Several approaches have been evaluated in the literature. In this paper, a decision tree based text-to-phoneme mapping is studied. A decision tree is trained for each letter according to information theoretic criteria on a pronunciation dictionary that contains the phoneme transcriptions for a large number of words. Context information is utilized to create the mapping. In our experiments, the mapping was constructed on the Carnegie Mellon pronunciation dictionary [1]. The phoneme accuracy of the most effective mapping was 99% on the training set and 91% on the test set of the pronunciation dictionary. The mapping was also implemented in a speaker independent isolated word recognition system. The recognition rates in the clean and in the car noise test environment were close to the baseline recognition rates obtained with the correct transcriptions, when the training lexicon contained the test vocabulary. When the test vocabulary differed significantly from the training vocabulary, the mapping performed below our expectations.

1. INTRODUCTION

Relatively simple applications, such as name dialing, have been among the first speech recognition systems introduced in consumer devices. Inherently dynamic vocabulary is one of the special characteristics of name dialing that needs to be considered in application design. Speaker independent sub-word based algorithms provide a feasible alternative to speaker dependent systems currently in use. On one hand the implementation cost is higher, but on the other hand the user avoids the burden of training up to dozens, or even hundreds, of name tags. In this paper, we describe the design of a sub-word based speech recognition system from the pronunciation modeling point of view.

When a dynamic vocabulary is used, some kind of mapping from text to pronunciation, or from letters to phonemes, is usually needed. Text-to-speech systems have been utilizing text (or grapheme) to phoneme mappings as supplements to pronunciation dictionaries. The fundamental problem is the same for speech recognition, and therefore, the same methods are applicable. The size of the mapping is important in embedded applications, and the choice between alternative algorithms is usually based on a compromise between the size of the model and its performance. The sensitivity of speech recognition systems to modeling errors varies according to the application vocabulary and the robustness of the acoustic models. When the mapping is created, it may not be possible to take into account all possible names encountered when the application is used. Hence the generalization ability of the mapping is also important. It is apparent that proper nouns are often very difficult to model since they do not always follow pronunciation rules. Therefore, special care must be taken to test the system on the same kind of words as the application is expected to use.

Several approaches have been used for text-to-phoneme mapping. A conceptually simple option is to try to create a set of pronunciation rules. In many languages, the task is quite simple, but in some languages it is very complicated. Pronunciation rules for the English language have been derived by, e.g., Meng et al. [2]. Their system utilizes detailed linguistic knowledge in order to find the text-to-phoneme rules. Dynamically expanding context for learning the text-to-phoneme mapping based on a pronunciation dictionary has been investigated by Torkkola [3]. His system automatically learns the mapping from the training data by considering the context in which the letter occurs in the orthographic form of the word. Pagel et al. [4] have used decision trees for providing a direct mapping from a text string to a corresponding phoneme sequence. The mapping was trained on an aligned pronunciation dictionary. The strength of decision trees lies in the ability to learn a compact mapping from a training lexicon by using information theoretic principles. In other words, they are efficient tools for lexicon compression. Neural network based methods attempt to achieve the same goal by different means (e.g., Andersen et al. [5]). Several approaches are also evaluated by Deshmukh [6], where, among other things, he describes a surname pronunciation database created by Mississippi State University. We chose to implement a decision-tree based system similar to the one presented in [4], because a well proven training algorithm exists and the implementation of the decoder is straightforward. The main objective of our work was to experiment with the algorithm in the context of speech recognition in a few different scenarios.

The rest of the paper is organized as follows. We present an alignment procedure first. Second, we describe the principle of using decision trees to generate text-to-phoneme mappings. Description of the evaluation of the performance of the mapping methods follows, both on pronunciation dictionaries and on a speech recognition task. Finally, conclusions are presented.

2. ALIGNMENT PROCEDURE

The text-to-phoneme mapping is trained on a pronunciation dictionary whose entries are composed of the spelling of words and their phonetic transcriptions. In the English language, for example, a letter may correspond to zero, one, or two phonemes. Therefore, as the first step, the phonetic transcription and the spelling of the entries in the pronunciation dictionary need to be aligned. The alignment is obtained by inserting graphemic or phonemic epsilons (nulls) between the letters in the text string, or between the phonemes in the phoneme sequence [4], [5]. The use of grapheme epsilons can be avoided by introducing a short list of pseudophonemes that are obtained by concatenating two

phonemes that are known to correspond to a single letter (e.g., /k_s/ for 'x').

In order to align the pronunciation dictionary, the set of allowed phonemes has to be defined for each letter. The phoneme list includes the pseudophonemes for the letter and the possible phoneme epsilon. These lists are the only language dependent part of the text-to-phoneme mapping algorithm. The initialization of the penalties used in the alignment is based on the correspondence between the letters and phonemes. For a given letter-phoneme pair, the penalty value is initialized with zero if the phoneme can be found in the list of allowed phonemes for the letter, otherwise it is initialized with a large positive value.

Given the initial penalty values, the pronunciation dictionary is aligned in two steps using the Viterbi algorithm. In the first step, all the possible alignments (producing the minimum score) are generated for each entry in the dictionary. Based on all the aligned entries, the penalty values are then re-scored. In the second step, only a single best alignment is found for each entry. These alignments are later used to train the actual text-to-phoneme mapping implemented with decision trees.

3. DECISION TREES

The mapping from the spelling of the word to the corresponding phoneme sequence is carried out using decision trees [4], [7]. The decision trees are trained on the aligned pronunciation dictionary. A single tree is trained for each letter which allows for training several trees in parallel, therefore speeding up the training phase.

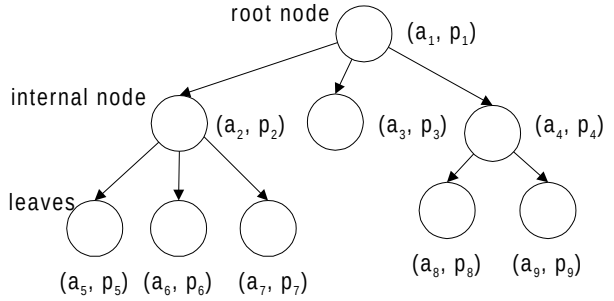

Figure 1: Exemplary decision tree showing nodes and leaves with attributes a_i and phoneme information p_j.

A decision tree is composed of nodes that are linked together as seen in Figure 1. An attribute is attached to each node. The attribute specifies what kind of context information is considered in the node. The context information may include the letters on the left and right hand side of the current letter. In addition, the previous phonemes and their phoneme classes may be used. Each node of the tree is followed by child nodes, unless the node is a leaf. We climb from a node to a child node based on the values of the attribute specified in the node. In order to find the phoneme transcriptions for words, the phonetic information needs to be stored in the tree. Therefore, in addition to the attribute, the phoneme that is most frequently encountered in the training data in the respective context is stored in every node.

When the decision tree is used for retrieving the phoneme that corresponds to the letter in a certain context, the search starts at the root node. The tree is climbed until a leaf is found. The phoneme that corresponds to the letter in the given context is stored in the leaf.

3.1. Training of decision trees

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 and phoneme contexts and the corresponding phoneme in the aligned pronunciation dictionary. 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. The splitting continues until the optimization criterion cannot be further improved.

In training, the root node of the tree is split first. In order to split the node into child nodes, an attribute has to be chosen. All the different attributes are tested and the one that maximizes the optimization criterion is chosen. Information gain [7] is used as the optimization criterion. In order to compute the information gain of a split, the phoneme distribution before splitting the root node has to be known. Based on the phoneme distribution in the root node, the entropy E is computed according to

$$E = -\sum_{i=1}^{N} f_i \log_2 f_i \,, \tag{1}$$

where f_i is the relative frequency of occurrence for the i^{th} phoneme, and N is the number of phonemes. Based on the letter and phoneme contexts, the training cases in the root node are split into subsets according to the possible attributes. For an attribute, the entropy after the split, E^s, is computed as the average entropy of the entropies of the subsets. If E_j^s denotes the entropy of the subset j after the split, the average entropy after the split is

$$E^s = \sum_{j=1}^{K} \frac{|S_j|}{|S|} E_j^s \,, \tag{2}$$

where $|S|$ is the total number of training cases in the root node, $|S_j|$ is the number of training cases in the j^{th} subset, and K is the number of subsets. The information gain for an attribute is given by

$$G = E - E^s \,. \tag{3}$$

The information gain is computed for each attribute, and the attribute that has the highest information gain is selected.

The splitting of the nodes in the tree is repeated for the child nodes. The training cases belonging to each child node are further split into subsets according to the different attributes. For each child node, the attribute that has the highest information gain is selected. The splitting of the nodes in the tree continues while the information gain is greater than zero and the entropies of the nodes can be improved by splitting. In addition to the information gain, the splitting is controlled by a second condition. A node can be split only if there are at least two child nodes that will have at least a preset minimum number of training cases after the split. If the information gain is zero or the second condition is not met, the node is not split.

3.2. Pruning of decision trees

Pruning is introduced in order to remove those nodes from the tree that do not contribute to the modeling accuracy. In addition, when pruning is applied the number of nodes, and the required memory for storage of the decision tree is reduced. Pruning can be done by introducing a stopping criterion into the training algorithm. Another possibility is to allow the decision tree to grow on the training set, and apply pruning on a pruning data set after training. Here the latter approach is taken since growing and pruning is reported to be more reliable in [7].

Error based pruning [7] is applied after the training step. A sub-tree is truncated to a leaf, if the predicted error rate at the root of the sub-tree is smaller than or equal to the predicted error rate at the leaves of the sub-tree. In addition, the sub-tree can be replaced by its most common branch if the predicted error rate of the sub-tree is greater than or equal to the predicted error rate of the most common branch. Pruning proceeds from the leaves towards the root node. For each node, the predicted error rate is computed. Also the predicted error rates of the child nodes and the most frequent branch are computed. If the predicted error reduction is greater when the sub-tree is replaced by its most frequent branch, the sub-tree is pruned accordingly. Otherwise the sub-tree is truncated to a leaf provided that the predicted error rate at the root of the sub-tree is smaller than or equal to the predicted total error rate in the leaves of the sub-tree. In our case the predicted error rates that are used in pruning are computed directly on the pruning data set.

3.3. Generation of phoneme sequences

After the decision trees have been trained for every letter, they can be used for generating phoneme sequences for the words in the vocabulary. For each word, a phoneme sequence is generated by going through the word from left to right one letter at a time. The decision tree corresponding to the letter in question is selected. Based on the context information the tree is climbed starting at the root node until a leaf is found. The phoneme that corresponds to the letter is stored in the leaf. Then the process moves on to the next letter and the phoneme that corresponds to it is found in a similar way. The phoneme sequence is constructed by concatenating the phonemes that have been found. Phoneme epsilons are removed from the sequence and pseudo-phonemes are expanded into distinct phonemes.

4. EXPERIMENTAL RESULTS

The text-to-phoneme mappings were trained and evaluated with the Carnegie Mellon University (CMU) American English pronunciation dictionary [1] with more than 115,000 entries. The dictionary was split into separate training and test data sets. The training set contained 80% of the entries and the test set covered the rest. Pruning was applied as described in the previous section. In the first test, all the names in the vocabulary were included in the training data set. In the second test case, the CMU pronunciation dictionary was used, but all the names in the recognition vocabulary were excluded from the training data set and they were placed in the test data set. Hence, the generalization capability of the mapping was crucial in this test. The MSU surname database (described in [6]) was evaluated for decision tree training in the third test. The main objective was to test how well the mapping can be trained on a training lexicon that is relatively small, but matches the test vocabulary well.

The performance of the text-to-phoneme mappings was also tested with an American English isolated word speech recognition system. The front-end of the recognizer produces standard Mel-Frequency Cepstral Coefficients (MFCCs), except that feature vector normalization has been included in the front-end to improve the noise robustness of the system as described in [8]. The back-end of the recognizer is based on the Hidden Markov Model (HMM) approach. The Viterbi decoder was implemented according to the token passing scheme. The acoustic models that are used by the back-end were trained on the American English TIMIT database. The phoneme models have sixteen mixtures per state. The total number of the HMMs in the decoder is 39. The recognition performance of the system was tested on an in-house American English speech database. The total number of speakers in the database is 23, and each of them uttered the test list of 80 names twice. The size of the test vocabulary was 1000 words. In addition to the clean environment, the system was tested in noise with the signal-to-noise ratio (SNR) varying between 5 and 15 dB. The noisy waveforms were obtained from the clean ones by artificially adding car noise at the desired SNR level.

4.1. First test case

The objective of the first test was to determine the performance of the text-to-phoneme mapping using the CMU lexicon. Moreover, its performance was evaluated in the speech recognition test described above, where the recognition test vocabulary (80 names; the first and last name included) was included in the training set. Table 1 shows that the performance of the decision trees is very good in a closed test, but it degrades on the test set, i.e., when material from outside the training set is used. Increasing the context length beyond 4 was not meaningful both from performance gain and storage points of view (the required space for storing the trees is roughly linearly proportional to the number of nodes). Both left and right contexts were used for letters. Only the left context was used for phonemes and phoneme classes.

Table 1: The phoneme accuracy and the number of nodes for the decision trees trained on the CMU pronunciation dictionary.

Context	Ph. acc. on training set [%]	Ph. acc. On test set [%]	Number of nodes
2	94.2	89.7	83476
3	97.6	90.6	118237
4	99.0	90.8	135334

Table 2: The recognition rates for the decision trees trained on the CMU pronunciation dictionary. The test vocabulary was included in the training set. The baseline rate was obtained using accurate transcriptions.

Context	Clean (%)	[+5…+15] dB (%)
Baseline	94.5	78.2
2	90.2	71.9
3	90.3	72.1
4	93.6	76.8

The speech recognition performance of the mappings is illustrated in Table 2. The performance of the generated phoneme sequences approaches the performance of the hand-made sequences when the context length increases.

4.2. Second test case

In the second test, we evaluated the generalization capability of the mappings in the previously described speech recognition test. The test names were excluded from the training set in this test. The phoneme accuracy remained virtually the same in the CMU tests since only a minor change was done from the point of view of the training process. However, there was a significant degradation in the performance of the mappings in the speech recognition test, as shown by Table 3.

Table 3: The recognition rates for the decision trees trained on the CMU pronunciation dictionary. The test vocabulary was excluded from the training set. The baseline rate (same as above) was obtained using accurate transcriptions.

Context	Clean (%)	[+5…+15] dB (%)
Baseline	94.5	78.2
2	83.9	64.2
3	81.4	61.6
4	84.4	63.6

When Table 2 and Table 3 are compared, it is quite evident that the generalization capability of the mapping to handle proper names is not satisfactory. It could be speculated that generic nouns follow pronunciation rules better, but in many applications the vocabulary is similar to our test case. Nevertheless, a real world system would be most likely trained on a lexicon that could be expected to cover the majority of names, so its performance is likely to be closer to the results presented in Table 2.

4.3. Third test case

The MSU surname pronunciation database (more than 18,000 entries, almost fully covered by the CMU lexicon) was also evaluated for the training of text-to-phoneme mappings. Since the training lexicon is much smaller, the resulting trees will be smaller as well. The expectation was that a training lexicon composed of proper names, even with a smaller number of entries, would be sufficient to train a mapping for name recognition. It turned out that even the tree with context length 4 (32,412 nodes) produced only 84.0% recognition rate in clean, and 61.2% rate in the noisy test case. These rates are similar to the ones obtained in the second test case. The mapping trained on the CMU lexicon contains generic language knowledge in addition to names (and also more proper nouns), which is likely to contribute to the performance gap.

5. CONCLUSIONS

We presented a method for generating phonetic transcriptions from text input. Applications, where the vocabulary is dynamic and where the storage of extensive pronunciation dictionaries is not possible, require a method that works with an open vocabulary. Several different approaches have been proposed in the literature. A very promising method is based on the utilization of decision trees. We evaluated decision trees using the Carnegie Mellon University pronunciation dictionary, where close to 99% phoneme accuracy was achieved in a closed test. We also evaluated the performance of the text-to-phoneme mapping on a speech recognition task. When the test names were included in the training lexicon, the performance was very close to baseline. However, a significant increase in the error rate was observed when the names were excluded from the training set. Also the results obtained with a smaller proper name database were significantly worse. It seems that the decision tree based method is very suitable for applications where the vocabulary is more or less known beforehand, or adheres to the pronunciation rules of the language. Alternative methods should be considered when lots of exceptions are expected.

6. REFERENCES

1. Weide, R.L., Carnegie Mellon Pronouncing Dictionary, Release 0.4, http://www.cs.cmu.edu.
2. Meng, H., Hunnicutt, S., Seneff, S., Zue, V., "Reversible Letter-to-Sound Generation Based on Parsing Word Morphology," IEEE Transactions on Speech and Audio Processing, 6, 1996, pp. 47-63.
3. Torkkola, K., "An Efficient Way to Learn English Grapheme-to-Phoneme Rules Automatically," Proceedings ICASSP 93, Minneapolis, Minnesota, 27-30 April, 1993, Vol. 2, pp. 199-202.
4. Pagel, V., Lenzo, K., Black, A.W., "Letter to Sound Rules for Accented Lexicon Compression," Proceedings ICSLP 98, Sydney, Australia, 30 November-4 December 1998, Vol. 5, pp. 2015-2018.
5. Andersen, O., Kuhn, R., Lazaridès, A., Dalsgaard, P., Haas, J., Nöth, E., "Comparison of Two Tree-Structured Approaches for Grapheme-to-Phoneme Conversion," Proceedings ICSLP 96, Philadelphia, PA, 3-6 October 1996, Vol. 3, pp. 1700-1703.
6. Deshmukh, N., "Maximum Likelihood Estimation of Multiple Pronunciations for Proper Nouns," Ph.D. Thesis, Mississippi State University, Mississippi, 1999.
7. Quinlan, J.R., "C4.5: Programs for Machine Learning," Morgan Kaufmann Publishers Inc., San Mateo, CA, 1993.
8. Häkkinen, J., Viikki, O., Tian, J., Vasilache, M., "Development of Robust Speaker Independent Command Word Recognition for Car Hands-Free Applications," Proc. IEEE Workshop on Robust Methods for Speech Recognition in Adverse Conditions, Tampere, Finland, 25-26 May 1999, pp. 139-142.