Designing very compact decision trees for grapheme-to-phoneme transcription

Anne K. Kienappel, Reinhard Kneser

Philips Research Laboratories
Weißhausstraße 2
D-52066 Aachen, Germany
{Anne.Katrin.Kienappel, Reinhard.Kneser}@philips.com

Abstract

Decision trees are a popular technique for automatic generation of a phonetic transcription for a given word spelling. We investigate different methods of decision tree design to obtain more compact trees and at the same time better grapheme-to-phoneme transcription quality. We evaluate different approaches to decision tree question selection and pruning using one English and two German grapheme-to-phoneme transcription tasks. In particular, we present a method of automatic generation of decision tree questions from the training data that significantly improves decision tree design.

1. Introduction

Grapheme-to-phoneme (G2P) transcription is required in automatic speech recognition as well as speech synthesis systems. The work presented in this paper has got a speech recognition application background. However, the techniques discussed are equally applicable in decision tree based G2P transcription systems in speech synthesis.

Our G2P transcription system is designed for on-line integration of new words into embedded automatic speech recognition systems. In the context of embedded systems, the G2P transcription is typically required to be compatible with tight constraints of computational resources, while maintaining a standard of high transcription quality. Some systems based on pronunciation rules compiled by phonetic experts may fulfill these conditions. However, great effort and expertise are required when setting up such a system for a new language or task. To achieve high flexibility, a system that can learn pronunciation rules from a training database is preferable. Decision tree technology [1] offers the opportunity to make small, flexible systems of good quality and is therefore a popular choice for G2P transcription (see e.g. references [2], [3], [4], [5]).

With an embedded system application scenario, our main objectives for decision tree design are not only to achieve optimal transcription quality, but also to minimise the tree size and thereby use of computational resources. It is well known that initial question selection and selecting the right subtree of the maximally expanded tree (pruning) critically affects both size and generalisation ability of the system. We have therefore conducted detailed investigations how question selection and pruning can be used to improve decision tree design with respect to our aims. We have achieved increased accuracy as well as considerable tree size reduction for three different tasks: German street names, UK-English mixed names and German general words.

Our paper is organised as follows. We start by outlining our G2P alignment method as well as specifying our node splitting criterion. We then discuss different types of G2P decision tree questions, in particular automatic generation of context questions about grapheme and phoneme groups. We also briefly introduce different methods of decision tree pruning. We finally report experimental results and draw conclusions. Note that we do not give a general introduction to G2P transcription using decision trees in this paper. Such introductions can can be found e.g. in references [3] or [5].

2. G2P alignment and splitting criterion

The main input for G2P decision tree training is a phonetic dictionary (lexicon), which is aligned into sequences of mappings of 1 grapheme to n phonemes, where n can be zero. To generate this alignment, we use a dynamic programming algorithm. The algorithm uses a manually compiled list of allowed $n \rightarrow m$ G2P mappings and imposes penalties on all non-allowed mappings. In a second dynamic programming step using a list of allowed $1 \rightarrow n$ mappings the $n \rightarrow m$ alignments are broken down into $1 \rightarrow n$ alignments. We prefer this semi-automatic alignment method to a fully automatic alignment (e.g. using an EM algorithm) because it allows easy identification of mistakes and exceptions in the training data and their exclusion from training on demand.

One grapheme is in our system typically one character; only few frequent character combinations that almost always correspond to a single phoneme (e.g. English ck or gh) are treated as a single grapheme. The spelling of words is parsed into a grapheme sequence using a longest match parse with the grapheme inventory, which is compiled by hand.

The criterion we use for splitting decision tree nodes is the standard minimum entropy criterion. If $N_{l,p}$ is the number of G2P mappings to phoneme sequence p in leaf l and N_l is the total number of mappings in leaf l, it is given by

$$\text{Entropy} = -\sum_{\text{leaves } l} \sum_{\text{phoneme sequences } p} N_{l,p} * \log(N_{l,p}/N_l). \quad (1)$$

3. Decision tree questions

A crucial part of decision tree design are the questions that are provided for node splitting. Choosing the wrong questions can lead to poor generalisation properties as well as excessive tree sizes.

The most standard G2P decision tree questions are those that specify single graphemes at a specific context position (*single grapheme* questions), and questions asking where a context position is located with respect to the word boundary (*boundary* questions). We also sometimes use *double* questions, asking "is

the grapheme at position X followed by an identical grapheme?" in order to make it easier for the system to learn some formal pronunciation rules, such as the German rule that short vowel are followed by double consonants.

In addition, phonetic context questions of the type "is phoneme X at position Y?" (*single phoneme* questions) can be included. Obviously, only phonemes at previously transcribed context positions can be taken into account, i.e. either right or left context according to forwards or backwards transcription. Phoneme questions have been reported to reduce error rates, e.g. by up to 12% relative for an English grapheme-to-(phoneme+stress) transcription task [4]. We want to quantify the usefulness of phoneme questions for G2P tasks without stress.

Finally, we can ask questions about groups of graphemes or phonemes (*group* questions, e.g. "is the phoneme at position -1 on of X, Y, Z?") instead of just single grapheme or phoneme questions. It is easy to see that group questions can lead to substantial reduction in tree size. In addition, they have a potential to lead to better generalisation behaviour. Group questions based on phonetic classes are used in various systems, e.g. [5] and [3]. In reference [3] they are reported to lead to small improvements ($\leq$ 2% relative word error reduction). However, we find that the relevance of phonetic classes to the G2P task is not immediately obvious. In addition, grapheme classification with respect to phonetic classes is not well defined. For these reasons we automatically generate group questions using criteria directly related to the G2P task.

3.1. Automatic group question generation

A clustering algorithm for group question generation can be found in reference [2]. Our automatic group question generation is based on this algorithm, but differs from it in several important aspects. It uses a maximal mutual information cluster criterion [2], which is in this context equivalent to entropy minimisation across clusters (replace "Leaf" by "Cluster" in equation 1). The basic top-down clustering algorithm for graphemes or phonemes from reference [2] can be sketched (here for the case of graphemes) as follows:

```
Initialise one cluster with all graphemes;
while( clusters with > 1 member ) {
        for all clusters with > 1 member {
                do bottom-up clustering to divide cluster;
        }
```

This includes the following *bottom-up clustering* routine:

```
Initialise single-grapheme clusters;
while( number of clusters > 2 ) {
        implement best possible merge;
}
move members between clusters to minimise entropy;
```

In the remainder of this section, we will specify some critical details regarding this algorithm and the use of group questions. We found that questions generated almost exactly as described in reference [2] reduced tree size but did not improve generalisation properties. However, with the modifications in group question generation and handling that are described below, we have obtained G2P decision trees that are both smaller and better generalising than trees with only single grapheme and phoneme questions.

We have so far described our algorithm as if graphemes or phonemes were clustered as abstract entities. However, the relevant units in an aligned training lexicon are G2P mappings.

type	context	members
grapheme	right	a,ae,ai,au,e,ea,ei,i,o,oi,u,y
grapheme	right	',b,c,ch,ck,d,f,g,gh,h,j,k, l,m,n,p,q,r,s,t,th,v,w,x,z
grapheme	right	k, q
phoneme	left	D, T, d
phoneme	right	D, T, f, v

Table 1: Examples of grapheme and phoneme groups automatically generated from the English names training corpus

What exactly then do we mean by a "grapheme" or "phoneme" X? Possible choices are all mappings with

- (for graphemes) X at position 0 (as in [2]).
- X at position Y for group questions about position Y.
- X at position -1 / +1 for left / right context questions.

We tried the first approach for grapheme group question generation without achieving any improvement in generalisation properties. The second approach uses intuitively more relevant features and makes sense for graphemes as well as phonemes. However, we observed that the correlations between graphemes at far-away context positions and output phonemes are very noisy and the groups which are created using it are rather random. We settled on the third approach as a good compromise. The grapheme and phoneme groups produced using this method are often surprisingly closely related to phonemic classes; some examples are listed in Table 3.1.

The top-down clustering algorithm does not yield a set of grapheme or phoneme groups as such, but a grapheme or phoneme tree. A simple example for such a tree could have a root node containing all graphemes (A, B, C, D), intermediate nodes (A, B) and (C, D), and the single graphemes (A), (B), (C), (D) as leaves. Several choices of grapheme groups could be derived from such a tree:

1. one group per generation below root level to make a small, complete[1] feature set [2], e.g. (A, B) and (A, C).
2. an incomplete subset of groups from intermediate levels, e.g. (A, B) or (C, D).
3. all groups, i.e. (A, B), (C, D) , (A), (B), (C) and (D).

Experiments showed that the first approach leads to poor generalisation properties, because graphemes that are not very similar (such as A and C in our example) are grouped together. We observed that for far-away context positions some coarse group questions, as in (2.), are sufficient; more detailed questions can even be harmful because they have a tendency to lead to excessive adaptation to the training data. However, for the closest contexts, a large set of very detailed questions, including single grapheme and phoneme questions, proved to be necessary. We therefore use all groups, as in (3.), up to a context length of ± 2. The level of detail up to which questions are included is decreased continuously for increasing context length. For the the maximum context length, only six questions, which describe the tree at the second branching level, are used.

The last point we want to discuss relates directly to the training algorithm. With a large choice of group questions, there are often several that achieve the same split for a certain node. Between questions that lead to equal gain in the splitting criterion, we use the following order of preference:

[1] "Complete" means that every group member can be isolated by taking the cross-section of a combination of groups or their complements.

1. Prefer closer context.
2. Prefer smaller grapheme/phoneme group.
3. Prefer non-phoneme questions.

While point one is intuitively obvious, point two reflects our observation of a tendency to falsely generalise exceptions if larger groups rather than smaller groups are preferred. The last point reflects the fact that phoneme questions build on unsafe features, because any previously transcribed phonemes may already contain mistakes.

4. Pruning and iterative cross-validation

To avoid trees that are unnecessarily large and excessively adapted to the training data, decision tree training algorithms typically choose an appropriate sub-tree of the fully trained tree. Of different pruning criteria that can be used to limit decision tree size, we tried the following

- Implicit pruning using stopping criteria: maximum no. of nodes, minimum gain in splitting criterion (global) or minimum node size (local).
- Pruning to minimise a cost-complexity "No. classification errors + $\alpha\times$ number of leaves" [1].

We also tried a decision tree design algorithm based on cross-validation by Gelfand, Ravishankar and Delp [6] (see also reference [3].) The tree is designed by iterative growing and cost-complexity pruning (complexity penalty α = 0) with two halves of the available training data, a process with guaranteed to converge. In our experiments the division of data in two roughly equal halves is achieved by randomly assigning each $1 \rightarrow n$ G2P mappings to one of two sets.

5. Experiments

5.1. Databases

An important aspect of G2P experiments is the overlap between training and test data. If the system is tested on words that were part of the training material, the experiment allows to examine the tree's capacity for lexicon *compression*. This is a relevant aspect for our application scenario in which there is typically no access to a large background lexicon. However, the harder and more important task for a G2P transcriber is the transcription of new words, which relies entirely on its *generalisation* abilities.

We have conducted experiments in three different domains: German street names, UK-English names and German words from a general background dictionary, which we will refer to as "German general". The German street names task is a typical mixed task of lexicon compression and generalisation. It uses two unrelated databases, a general street name collection for training and a collection of street names in Frankfurt(Main) for testing. For the other two tasks, we used block division of one database to prevent overlap between training and test. An X-Y-Z-Y block division of a lexicon means the fillowing. Starting from the first entry of the alphabetically ordered lexicon, we put X entries into the training lexicon, discard the next Y entries, put Z entries into the test lexicon, discard Y entries and so on. The English names task is designed to test mostly the generalisation ability of the transcriber. The German general task is, despite discarding alphabetical word blocks via 80-8-4-8 block division, a mixture of both compression and generalisation. Here, the training/test overlap consists in constituents of composite words that do not occur in the same alphabetic block (e.g. the word "Boot" in "Hausboot" and "Segelboot").

	German street names	UK-English names	German General
Training lexicon	Mixed street names	80-0-20-0 block div.	80-8-4-8 block div.
size	11k	74k	320k
Test lexicon	Frankfurt street names	80-0-20-0 block div. block div.	80-8-4-8
size	8k	19k	16k
overlap training/test	30% of test	none	via composita

Table 2: G2P training and test database information.

	SER	PER	size (leaves)
Ger. streets	14.5% (-5%)	1.5% (-5%)	1.5k (-21%)
En. names	33.7% (-10%)	8.5% (-5%)	3.7k (-58%)
Ger. Gen.	8.5% (-8%)	1.0% (-1%)	9.7k (-63%)

Table 3: Error rates and size of best trees (with respect to PER) and relative reduction with respect to best trees in the base line system (grapheme and boundary questions only)

More information about our databases, which were compiled using a combination of various sources, is given in Table 2. All lexica were filtered to exclude numerals and abbreviations (identified by capital letters at positions other than the beginning of the word). For the two names tasks, the training data were reduced by $\approx$ 1% by excluding entries that could not be aligned using allowed $n \rightarrow m$ mappings.

We measure performance by counting transcription errors on two levels: on phoneme level (phoneme error rate or PER) and on the level of lexicon entries (string error rate or SER). Note the difference between SER and a transcription word error rate for lexicon entries with multiple words. If a street name lexicon has one entry "Reading Road" and the word "Reading" has got a transcription error, then we count a SER of 100%, while a transcription word error rate would be 50%. There are only negligible qualitative differences in the experimental results if the performance is measured in of SER or PER. We made the (rather arbitrary) decision to plot our results in terms of string error rates and quote the PER only for single data points.

5.2. Results

We have plotted selected experimental results regarding question selection, pruning and cross-validation in Figure 1. For these experiments we used the following basic settings, which were previously optimised with respect to generalisation properties: The maximal context was 4 for the German and 5 for English tasks. Double questions were included only for the German tasks. German systems with phoneme questions use forwards transcription, English ones (in agreement with results from reference [4]) backwards transcription.

The consistent improvements in accuracy and reductions in tree size achieved by improving the question inventory can be seen in Figure 1. A summary of the total improvement achieved by the use of phoneme and group questions is given in Table 3. The detailed results in Figure 1 show that both kinds of questions contribute to the total improvement, to varying degrees for the different tasks.

We verified for the names tasks that cost-complexity pruning yields significantly better results than implicit pruning using stopping rules. Surprisingly, iterative cross-validation for decision tree design yields trees that perform significantly worse than trees of the same size obtained using simple cost-complexity pruning. However, the English names task shows that the cross-validation method yields a good estimate of the optimal tree size. For the German tasks, we cannot expect such a result because training and test set overlap significantly. For German street names we measured generalisation ability by extrapolating from the data in Figure 1 and error rates measured on the training data in order to obtain an estimate of the error rates on "new" words. We found that the tree size given by cross-validation agreed well with the size of best generalisation ability.

6. Conclusions

We have investigated several methods of decision tree design for automatic G2P transcription. In addition to optimisation of transcription accuracy, we tried to minimise tree size to suit the tight computational constraints that are typical for embedded automatic speech recognition systems. We have tested these methods on two German and one English transcription tasks.

We have verified that questions about the phonetic context can significantly improve transcription accuracy. The use of grapheme and phoneme group questions, which are automatically generated from the training data by means of a maximum mutual information clustering algorithm, yields further improvements. Phoneme and group questions together lead on average to a relative reduction of 8% in error rate on the level of lexicon entries as well as a tree size reduction by roughly half.

The best tree design was achieved by cost-complexity pruning using all training data. In particular it yielded error rates that were significantly lower than those achieved with iterative cross-validation training [6]. However, the latter yielded good predictions of the tree size that optimizes generalisation properties.

7. References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone "Classification and regression trees", Belmont, CA: Wadsworth, 1984.

[2] Lucassen, J.M. and Mercer, R.L., "An information theoretic approach to the automatic determination of phonemic baseforms", Proc. ICASSP, 42.5.1 – 42.5.4, San Diego, CA ,1984.

[3] Andersen, O. and Kuhn, R. and Lazarides, A. and Daalsgard, P. and Haas, J. and Noeth, E. "Comparison of two tree-structured approaches for grapheme-to-phoneme conversion", Proc. ICSLP, p. 1700–1703, Philadelphia, PA", 1996.

[4] V. Pagel, K. Lenzo and A. W. Black "Letter to sound rules for accented lexicon compression" Proc. ICSLP, p. 2015–2018 Sydney, Australia, 1998.

[5] J. Suontausta and J. Häkkinenen "Decision tree based text-to-phoneme mapping for speech recognition" Proc. ICSLP, p.831-834, Bejing, China, 2000

[6] S. Gelfand, C. Ravishankar and E. Delp "An iterative growing and pruning algorithm for classification tree design", IEEE trans. PAMI, Vol. 13, p. 163-174, Feb 1991.

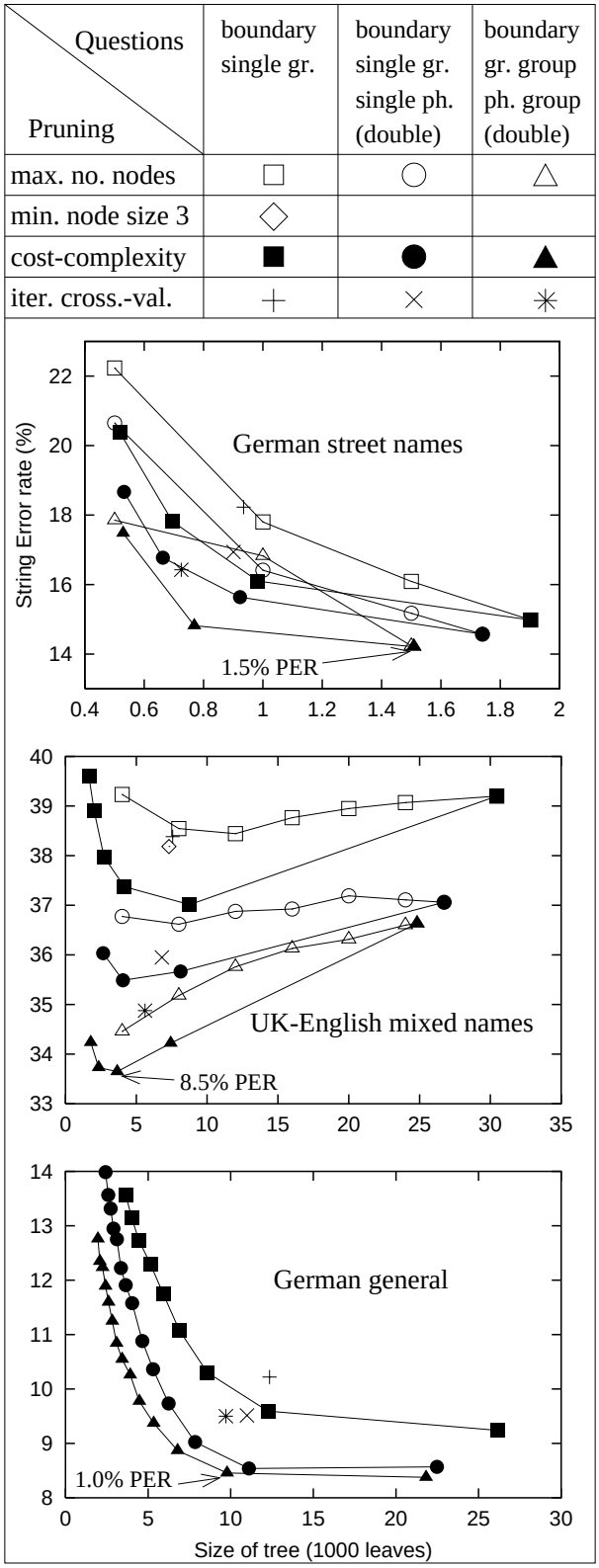

Figure 1: *String error rates depending on tree size for different pruning methods and different question inventories. "gr." and "ph." are short for "grapheme" and "phoneme". Double questions were used only for the German tasks.*