IMPROVEMENTS ON A TRAINABLE LETTER-TO-SOUND CONVERTER

Li Jiang, Hsiao-Wuen Hon and Xuedong Huang

Microsoft Research
One Microsoft Way
Redmond, Washington 98052, USA

ABSTRACT

Letter-to-sound (LTS) conversion is important for both text-to-speech (TTS) and automatic speech recognition (ASR). In this paper we discuss some improvements we have made on our trainable LTS converter. We use a classification and regression tree (CART) to automatically configure the most salient phonological rules needed for the LTS conversion. We address problems in growing multiple trees and use of phonotactic information for better generalization. The experiments were carried on both the NETTALK database and the CMU dictionary. With improved techniques, the conversion error rate at the phoneme level and word level was reduced by 15% and 20% respectively. For both tasks, the phoneme conversion error rate was reduced to about 8%.

1. INTRODUCTION

Letter-to-sound (LTS) conversion plays a very important role in both text-to-speech (TTS) and automatic speech recognition (ASR). LTS module is necessary for any TTS system in order to convert every word into a sequence of phonemes, as a specific word may not be in the system dictionary, no matter how big the dictionary is. For the same reason, ASR systems also needs LTS, as new words have to be added on the fly, either by the user or through interface from speech-aware applications.

The traditional rule-based LTS system requires tedious effort to improve its quality and a different language often requires a completely new set of rules. A trainable LTS converter is attractive since its performance can be improved by learning from existing dictionaries and it can be easily configured for different languages using available dictionaries.

Trainable LTS converters have been studied by a number of people [1-5]. Most of them used statistic approaches such as neural networks [3,4], Hidden Markov Models (HMM) [5] or the classification and regression tree (CART) [1,2]. Among these existing systems, CART-based LTS generated one of the most accurate systems.

In this paper, we extend the typical CART approach to generate more accurate LTS conversion. It is observed that the use of improved tree growing procedure, deleted-interpolation based smoothing, phonotactic information and multiple trees can substantially improve the performance for predicting unseen data. We have achieved overall error reduction of 15% at phoneme level and 20% at word level. On both NETTALK database and CMU dictionary tasks, the phoneme conversion error rate has been reduced to about 8%.

2. EXPERIMENTAL DATABASE

All of experiments were carried on two databases. The first one is the NETTALK [3], which has hand-labeled alignment between letter and phoneme transcriptions. The second one is the CMU dictionary [8], which does not have any alignment information.

The NETTALK database consists of 19,940 entries, we randomly selected 14,955 entries as training set and reserved the rest 4,951 words for testing. Notice that those 4,951 words correspond to 4985 entries in the database because of multiple pronunciations. The hand-labeled alignments were used directly to train the CART for LTS conversion.

The CMU dictionary has more than 100,000 words, we selected the top 60,000 words based on the unigram trained from North American Business News for this experiment. Among them, 52,415 entries were used for training and 9,719 words were reserved for testing. Due to multiple pronunciations, those 9,719 words have 10,520 entries in the dictionary. Note that some flap rules were applied to the dictionary to change some "D" and "T" phonemes in certain contexts to "DX". Due to lack of alignment information, we needed to run dynamic programming to align each letter to the corresponding phoneme before training the LTS CART.

We tested both word and phoneme level conversion accuracy for all of our experiments. For words with multiple pronunciations in the database, hitting one of the pronunciations was counted as correct in computing word accuracy. However, when phoneme accuracy was calculated,

all the pronunciations were used for comparison. There was no extra effort to match the multiple pronunciations against the LTS output candidates. Therefore, the phoneme accuracy reported here could be slightly improved with careful handling of multiple pronunciation entries. Note that all the "NULL" phonemes generated by LTS were always removed in computing the accuracy.

3. BASELINE

The CART [6] has been widely used in speech recognition [1,2,9] and language modeling [10] because of its flexibility and generalization capability. It is a top-down classification approach. The basic component includes a set of YES-NO questions and a procedure to select the best question at each node to grow the tree from the root.

The basic YES-NO question for LTS conversion looks like "*Is the second right letter 'p'?*" or "*Is the first left output phoneme 'AY'?*" The questions for letters could be on either left or right side. For phones, we only used questions on left side for simplicity. The range of question positions should be long enough to cover the long-distance phonological variations. In our empirical experiment, we found that the 11-letter window (5 for left letter context and 5 for right letter context) and 3-phoneme window for left phoneme context is generally sufficient.

A primitive set of questions would be the set of all the singleton questions about each letter or phoneme identity. When growing the tree, we simply chose the question that has the best entropy reduction at each node. We observed that if we allow the node to have complex question which is a combination of primitive questions, the depth of the tree will be greatly reduced and the performance will be improved. For example, complex question "*Is the second left letter 't' and the first left letter 'i' and the first right letter 'n'?*" can capture 'o' in common suffix "*tion*" and convert it to the right phoneme. Complex questions can also alleviate data fragment problem caused by greedy nature of the CART algorithm. Our way of finding such complex questions was similar to the ones used in [9,10].

We built the baseline system using the above techniques and the error rates are listed in Table 1.

Database	Phoneme	Word
CMU Dict	9.7%	35.0%
NETTALK	9.5%	42.3%

Table 1. LTS baseline

4. DISTANCE-WEIGHTED ENTROPY REDUCTION

It has been shown [2] that the question set consisting of only pure singleton questions will hurt generalization because of potential training data overfitting even with complex questions as mentioned in section 3. It was verified with our experiments that category questions would yield a tree for better generalization.

Category questions can be formed in both letter and phone domain with the help of some common linguistic knowledge. For example, the most often used set includes the letter or phone clusters for vowels, consonants, nasals, liquids, fricatives etc. Adding this new category question set into our singleton question set, we obtained slightly improved results in Table 2. It was suggested that those category questioned can be attained automatically by clustering training samples [10]. Unfortunately, our attempt did not lead to any improvement.

Database	Phoneme	Word
CMU Dict.	9.3%	33.0%
NETTALK	9.3%	41.3%

Table 2. LTS using category question set

One important finding in growing the decision tree was that the context distance plays a major role in the overall quality. It is very important to weight the entropy reduction according to the distance (either letter or phoneme) to avoid over-generalization. With limited training samples, using distant letter and/or phoneme question can result in simpler tree structure. But on the other hand, it might cause over-training. The generalization capability usually gets better when entropy reduction is discounted by its distance from the center. Intuitively, it forces the tree to look at the "nearby" context more carefully before the "far-away" contexts. The weights for entropy reduction were determined by empirical methods. The results in Table 3 were obtained by using category question set and distance-weighted entropy reduction.

Database	Phoneme	Word
CMU Dict.	8.7%	30.3%
NETTALK	9.1%	39.8%

Table 3. LTS using category question set and distance-weighted entropy reduction

5. DELETED-INTERPOLATION BASED SMOOTHING

Each leaf of the LTS CART has a probability distribution for letter to phoneme mapping. The deeper the tree, the sharper the distribution is. However, when the distribution is too sharp, one particulate phoneme will usually dominate, i.e., all the other phonemes will get near 0 probability. This will make LTS less robust and make it unrecoverable from a single mistake. There are two types of solution for the over-training problem: pruning or smoothing.

Pruning will control the depth of the tree. For example, certain criteria have to be met for a node to be split. Typically it requires a minimum number of counts and a minimum entropy reduction. Of course, a tree can also be

grown very deeply and then pruned back to the right size. Cross-validation generally are employed for CART pruning. The problem of pruning is that classification resolution is typically sacrificed.

Smoothing, on the other hand, does not change the structure of the tree. It will smooth the distribution at the leaves to avoid the over-fitting but also maintain the high classification resolution. For example, the leaf distribution can be interpolated with the distributions of its ancestor nodes. In addition, smoothing can be combined with pruning together.

Deleted-interpolation [7] has been widely used for computing the interpolation weights of maximum likelihood based statistical models. For LTS CART smoothing, we can partition the training data into two sets (or more), tie the leaf nodes into classes and use deleted-interpolation technique to compute the interpolation weights for each class. The class can be based on counts of leaf distributions, entropy of the leaf distributions, letter identity, phoneme identity or any combination of the above.

In our experiments, we found that interpolation with a series of ancestor nodes is better than with only the immediate parent node. There was one specific set of deleted-interpolation weights for each letter tree. The results show that it can greatly improve the N-Best coverage without sacrificing the top-one accuracy. For example, we were able to reduce the top-10 word error rate from 29.3% to 13.4% for CMU dictionary. The better N-Best coverage paved the road for combining other type of information for further performance improvement.

6. PHONEMIC TRIGRAM RESCORING

Even though there were questions on left phonemic context for the CART, we felt that we had not made full use of the phonotactic information. Statistical model such as phonemic trigram model, on the other hand, is more powerful on modeling phonotactic information and can provide probabilistic score for the occurrence of any phoneme sequence.

Database	Phoneme	Word
CMU Dict.	8.6%	28.2%
NETTALK	8.6%	36.5%

Table 4. LTS using phonemic trigram rescoring

Similar to speech recognition, we can use the phonemic trigram in the same way by combining the score from the LTS CART and phonemic trigram using a "language weight". In particular, we used trigram for rescoring of the N-Best list generated by LTS. The phonemic trigram was generated from the training samples with backoff smoothing. The results in Table 4 show the improvement by combining information from phonemic trigram rescoring.

7. MULTIPLE TREE COMBINATION

By examining the errors produced by LTS converter, we found that most of the errors were caused by over-generalization. It indicated that we could potentially improve the performance by partitioning data and building multiple trees. With different prediction capability, the errors could be recovered by combining results from multiple trees.

In a pilot experiment, we evenly partitioned the training data into two parts and trained two trees respectively. When we tested the performance of these two trees, we found that they had a great overlap but also behaved differently as each of them has different focus region. Combining them together greatly improved the coverage.

To get a better overall accuracy, we used the tree trained by all the samples together with two other trees trained by half of the samples respectively. The leaf distributions of three trees were interpolated together with equal weights and then phonemic trigram is used to rescore the N-Best output lists. Moreover, multiple trees can also be viewed as another smoothing mechanism. For example, the top-10 word error rate on CMU dictionary was further reduced from 13.4% to 10.9%. With multiple trees and trigram rescoring, the results shown in table 5 demonstrated that our LTS is clearly among one the most accurate ones reported on the same databases so far.

Database	Phoneme	Word
CMU Dict.	8.2%	26.9%
NETTALK	8.1%	34.2%

Table 5. LTS using multiple trees and phonemic trigram rescoring

8. SYLLABLE INFORMATION

We have tried to use syllable structure information to improve the conversion accuracy as syllable is a relatively independent unit. It is likely that the pronunciation of a syllable will be relatively independent of its contexts. If we incorporate the syllable boundary into the CART, we might potentially get better performance.

In the pilot experiment, we assumed that we knew the syllable boundary by using dictionary with syllable boundaries predetermined and marked. We then mapped the boundaries from phoneme domain to letter domain. In this case, we achieved a moderate performance improvement by asking questions on syllable boundary in tree generation.

One step further, we used a very simple rule-based syllabification algorithm to derive syllable boundary for a given

word. We assumed that we knew the correct pronunciation. This simple syllabification algorithm did make some errors when compared to dictionary with syllable marks. Even though we did not always use the correct syllable boundary information in the CART, it still demonstrated modest improvement but the gain was smaller compared to the perfect syllable boundary case.

Finally, we removed all the assumption and had a two-pass LTS converter. We used the output of the CART (without syllable information) and the rule-based syllabification algorithm to derive the syllable boundaries in the first pass and then redid LTS in the second pass using the obtained syllable boundary information. Unfortunately, the performance improvement became very minor.

Even though we have not achieved significant performance improvement with syllable structure information so far, we strongly believe that it contains information that is not currently used and has great potential to improve the LTS performance. Further investigation will need to be done on this subject.

9. ERROR ANALYSIS

After analyzing the errors made by LTS, we can classify them into a few categories. The first category goes to proper nouns and foreign words, which are the most difficult part of the LTS. For example, "Pacino" will be converted *to "P AX S IY N OW "* instead of "*P AX CH IY N OW*" because LTS does not know it is a foreign name and it should convert differently. The second category goes to generalization errors. There are two types of generalization errors. The first one is over-generalization of some partial letter sequence. For example, word "shier" would likely be converted to "SH IH R" instead of correct pronunciation "*SH AY R*" if word "cashier"("*K AE SH IH R*") appears in the training data. The other generalization error is over-generalization for patterns not appearing in the training data at all. Of course, there is no hard boundary between the first and second case. The final category is errors caused by inconsistencies in the transcription of the dictionary. For example, CMU dictionary transcribes the ending letter "s" in word "lamos" to "S" ("*L AA M OW S*") but transcribes it in word "alamos" to "Z" ("*AE L AX M OW Z*"). Of course, one might argue this type of errors should be database errors, rather than LTS errors.

To find out what kind of errors the LTS produces, we obtained the top 10 phoneme confusion pairs from our LTS output for CMU dictionary. They are IX/AX, DX/T, AE/AX, AA/AX, R/ER, S/Z, EH/AX, IH/IY, DX/D, AO/AA. Particularly, The top one confusion between IX/AX dominates the entire confusion sets. It should not be a surprise because IX/AX are among the most inconsistent transcriptions in any dictionary and there is almost no consensus for IX/AX transcription among different dictionaries.

10. SUMMARY

We have demonstrated that distance-weighted entropy reduction, deleted-interpolation based smoothing, phonemic trigram rescoring and multiple tree combination can collectively improve the generalization ability of CART-based LTS conversion. We have achieved 20% error reduction on word level conversion and produced one of the most accurate LTS converters that is practical, accurate, and trainable. The future work will include the study on stress assignment, how to get and use the syllable structure information and integration with a morphological component to achieve better robustness.

REFERENCES

[1] J.M. Lucassen and R.L. Mercer, "An Information Theoretic Approach to the Automatic Determination of Phonemic Baseforms", Proc. of ICASSP-84, 42.5.1 – 42.5.4, 1984

[2] O. Andersen, R. Kuhn, A. Lazarides, P. Dalsgaard, J. Haas and E. Noth, "Comparison of Two Tree-Structured Approaches for Grapheme-to-Phoneme Conversion", in Proc. of ICSLP 96, pp 1700-1703, 1996

[3] T.J. Sejnowski and C.R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text", Complex Systems, pp145-168, 1987

[4] M.J. Adamson and R.I. Damper, "A Recurrent Network that Learns to Pronounce English Text", in Proc. of ICSLP 96, pp 1704-1707, 1996

[5] S. Parfitt and R. Sharman, "A Bi-directional Model of English Pronunciation", in Proc. Eurospeech 91, pp. 801-804, 1991

[6] L. Breiman, J. Friedman, R. Olshen and R. Stone, "Classification and Regression Trees", Wadsworth International Group, Belmont CA, 1984

[7] F. Jelinek and R.L. Mercer, "Interpolated estimation of Markov Source Parameters from Sparse Data", in Proc. of the workshop on Pattern Recognition in Practice, North-Holland Publishing Company, 1980

[8] CMU Dictionary, Carnegie-Mellon University, 1995

[9] H.W. Hon, "Vocabulary-Independent Speech Recognition: The VOCIND System", PhD these, School of Computer Science, Carnegie Mellon University, February 1992

[10] L. Bahl, P. Brown, P. de Souza and R. Mercer, "A Tree-Based Statistical Language Model for Natural Language Speech Recognition. IEEE Transaction on Acoustics, Speech, and Signal Processing, vol. ASSP-37(1989), pp. 1001-1008