Phonetic Question Generation Using Misrecognition

Supphanat Kanokphara and Julie Carson-Berndsen

School of Computer Science and Informatics
University College Dublin,
Ireland
{supphanat.kanokphara, julie.berndsen}@ucd.ie

Abstract. Most automatic speech recognition systems are currently based on tied state triphones. These tied states are usually determined by a decision tree. Decision trees can automatically cluster triphone states into many classes according to data available allowing each class to be trained efficiently. In order to achieve higher accuracy, this clustering is constrained by manually generated phonetic questions. Moreover, the tree generated from these phonetic questions can be used to synthesize unseen triphones. The quality of decision trees therefore depends on the quality of the phonetic questions. Unfortunately, manual creation of phonetic questions requires a lot of time and resources. To overcome this problem, this paper is concerned with an alternative method for generating these phonetic questions automatically from misrecognition items. These questions are tested using the standard TIMIT phone recognition task.

1 Introduction

One of the main advantages of statistical speech recognition systems is that they are assumed to not to require a lot of language-specific linguistic knowledge; once an annotated corpus is available as acoustic training data, the system can be trained to build models from that data. However, since most current speech recognition systems are based on context-dependent Hidden Markov models (HMM), this requires a large number of context-dependent units to be trained. Unfortunately, no single corpus (or even multiple corpora) can possibly contain such a large number of units.
In order to alleviate this problem and strike a balance between the number of context-dependent units and the limited acoustic training data, tree-based state tying is commonly employed [1], which allows parameters that exhibit similarity to be shared between context-dependent units. The level of similarity is determined automatically from a phonetic decision tree.

For reasonable modeling, these shared parameters are constrained by a set of phonetic questions. These questions aim to determine the similarity of the contexts and often rely on the phonetic judgments of a human expert who can determine whether the contexts refer to similar contexts based on phonetic categories such as consonant, vowel or labial. In other words, this requires language-specific linguistic knowledge. To reduce the manual effort in the question construction procedure, there

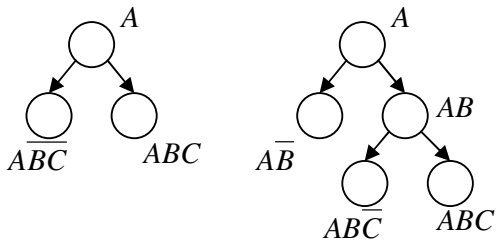

Fig. 1. Decision tree with or without intersection class question.

have been attempts to automatically generate questions for tree-based state tying systems [2], [3] and [4].

Even though these automatic systems differ, they have one thing in common; they generate phonetic questions by defining phone classes which have some similar properties. This is reasonable since experts also manually transcribe phonetic questions by mean of phone similarities. As a result, phonetic questions from automatic systems are usually as good as or only slightly worse than phonetic questions defined by experts. However, the benefit from automatic systems is significant. Phonetic questions can be generated quickly. This is useful for fast development of new language speech recognition systems.

Phone recognizers normally use the Viterbi search to map a sequence of phones to a speech utterance. Since the recognition result is not always perfect, misrecognized phones are inevitable. The motivation of this paper comes from the assumption that if a phone recognizer wrongly hypothesizes a phone as another phone, these confusable phones should have some level of similarity. Hence, these confusable phone classes can be used to generate phonetic questions.

According to [4], it is quite obvious that including combinations (intersections) of two or more classes in the phonetic question set yields higher system accuracy. This can be illustrated in Fig. 1. The node in the left tree is clustered with class BC while the nodes in the right tree are clustered with class B and C. Both trees yield the same results, which are the classes ABC. However, the difference is that the right tree must generate the classes $A\overline{B}$ and $AB\overline{C}$, which may not be suitable classes for tying. To generate all possible class combinations, another tree cluster is introduced. This tree is different from the conventional decision tree. Decision trees choose a class to cluster optimally depending on the likelihood score. This tree requires no likelihood score. It clusters all classes orderly. Decision trees stop clustering earlier when the criteria are met. This tree clusters each class until there is only one member in that class or there are no more classes left for clustering and therefore all possible class intersections can be generated.

Systematically, the paper is organized as follows. Section 2 lists a number of techniques for generating confusable phone classes. Section 3 shows how to generate class intersections from confusable phone classes. Section 4 shows the experimental result while section 5 draws the conclusion and future works.

Table 1. Example of misrecognition table

phones	/a/	/b/	/c/	/d/
a	10	1	0	0
b	3	12	1	0
c	0	2	20	5
d	0	0	0	6

2 Misrecognition

In this paper, confusable phone classes are generated using phone substitution errors hypothesized by context-independent acoustic models (phone deletion and insertion are ignored). Although the idea of using confusable phone classes as phonetic questions appears reasonable, the weak point of this approach is the same as other automatic phonetic question generation systems, namely that the quality of phonetic questions greatly depends on the quality of speech used to generate the questions. In order to fully use confusable phone classes as phonetic questions, a number of techniques are applied to reduce this error.

Firstly, all confusable classes are used directly. This is the simplest use of misrecognition. However, this may be risky because of out-of-class misrecognition errors. The second technique is called count-limited misrecognition. This technique comes from the assumption that the number of phones that are misrecognized out of class should be small and if the number of misrecognized phones is less than the threshold, that phone should not be counted in the class. Finally, a "cross constraint" technique is tested. For this technique, only two-way misrecognitions are accepted. For example, if "p" is recognized as "b" and "b" is recognized as "p", "p" and "b" are in the same class. However, if "a" is recognized as "l" while "l" is not recognized as "a", "a" and "l" are not in the same class.

3 Generation of Class Intersections

After the misrecognition classes are obtained, the class intersections can be generated by combining all of these classes. The concept of generating class intersections is simple. The algorithm starts from a class and cluster with other classes until there is only one member in the class or no more classes left for intersection. To clearly explain this, let us assume that a phone recognizer misrecognized phones as shown in Table 1. The number in each cell indicates how many times a row phone is recognized as a column phone. For example, "a" can be recognized as /a/ ten times, /b/ one time, /c/ and /d/ zero time. Classes from this table are (from each column) {"a", "b"}, {"a", "b", "c"}, {"b", "c"} and {"c", "d"}. {"a", "b"} means both "a" and "b" can be recognized as /a/. {"a", "b", "c"} means "a", "b" and "c" can be recognized as /b/ and so

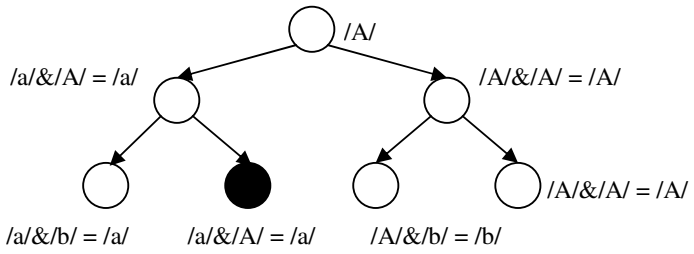

Fig. 2. Illustration of algorithm for two columns

on. Then, {"b"} and {"c"} classes can be generated from the class intersections /b/&/c/ and /c/&/d/, respectively.

The algorithm to generate these class intersections is as follows.

1. Read each column from the misrecognition table.
2. List all classes with the phones where the number of recognition is higher than a threshold, e.g. from table 1 /a/ {"a", "b"} (threshold = 0), etc.
3. Add a special class called the /A/ class to the class list. This class contains all phones in the table ({"a", "b", "c", "d"} in this case).
4. Use /A/ class as the root node of the tree.
5. For each column
   - 5.1. List all classes in the column. This includes /A/ class.
   - 5.2. For each activated leaf node in the tree.
     - 5.2.1. Split the node according to class list constructed in step 5.1.
     - 5.2.2. For each split node
       - 5.2.2.1. Find the intersection of classes between the node and its parent node. For example, node /c/ can split to /c/&/d/ node.
       - 5.2.2.2. If the node contains the empty set, deactivate the node.
       - 5.2.2.3. If the node contains the same phones as any node in step 5.2.1, deactivate the node.
6. All leaf nodes are confusable phone classes.

Fig. 2 shows the algorithm procedure. The black node indicates a deactivated node. In the figure, the node is deactivated according to step 5.2.2.3.

4 System Overview

4.1 Phone Recognition System

The phone recognition system used in this paper has been constructed using HTK [5]. All speech files are parameterized into 12 dimensional PLP, 0^{th} cepstrums and their deltas and accelerations (39 length front-end parameters). Flat start training is then

used for model initialization according to the gender-dependent phones. The transitions of male and female phones are tied together for robustness. Each model contains 5 states and the covariance matrices of all states are diagonal (left-right model with no skip state).

After context-independent HMMs have been trained, they are expanded to context-dependent HMMs using a cross-word network. Phonetic decision trees are then used to cluster the context-dependent HMM states into classes according to phonetic questions generated from the system in section 3. These classes are tied and trained together. From the context-dependent HMMs, the number of model mixtures is increased by 1 and the models are trained. This process continues until the number of mixtures is 10.

All training processes are estimated using the maximum likelihood algorithm. The number of training iterations after each change is determined automatically in line with [6]. The language model is trained from the phone sequences of the training set using back-off bigrams. For the recognition process, the Viterbi algorithm is used without any pruning factor.

4.2 The Corpus

The experiments use the standard TIMIT corpus [7] consisting of 6300 sentences, 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the U.S., of which 462 are in training set and 168 are in the testing set. There is no overlap between the training and testing sentences, except 2 dialect (SA) sentences that were read by all speakers. The training set contains 4620 utterances (326 males and 136 females) and the testing set contains 1680 (112 males and 56 females). The core test set, which is the abridged version of the complete testing set, consists of 192 utterances, 8 from each of 24 speakers (2 males and 1 female from each dialect region). In this paper, SA sentences are eliminated from the training set because they occur in both the training and testing sets. In this paper both core and complete test sets are used for evaluation. Automatic phonetic questions are generated from the misrecognition of the training set. In this paper, TIMIT original phone set is converted into traditional 39 phone set [8] before training.

4.2 Cheat Phonetic Question Set

To ensure that the quality of manually generated phonetic questions is sufficiently good (not an unfair experiment as a result of using poor quality handmade phonetic questions), a set of questions is first transcribed by a phonetician. These phonetic questions are constructed based on phone classes from a number of different sources [1], [9] and [10]. A number of phonetic questions are removed from the question set by trial-and-error until the highest system accuracy is obtained. This trial-and-error process is tested with the TIMIT core test set. This *cheat phonetic question set* is used as a baseline in this paper. This is called cheat phonetic question set because it is adjusted optimally for TIMIT core test set. Note that in reality, cheat phonetic questions

Table 2. Misrecognition types

Type	Core	Complete	# questions
direct	73.5	n/a	80,476
1-limited	73.2	n/a	24,008
cross constraint	73.8	73.8	23,852
baseline	74.4	74.0	868

are not possible to be generated because the test set is unknown. Therefore, the accuracy of general models should be lower than the models trained from cheat questions.

5 Experiment

The experiment in this paper is separated into two phases. The first phase tests a number of techniques as described in Sect. 2. The best technique is then selected and passed to the second phase. In the second phase, the models for generating misrecognition are altered. In the first phase all misrecognitions are generated from simple context-independent models while in the second phase, misrecognitions are generated from more complex models.

5.1 Phase One

Firstly, we tried to find the best misrecognition technique for generating phonetic questions. In this phase, misrecognitions are trained from context-independent models. This test is performed on the TIMIT core and complete test sets. Table 2 shows experimental results according to three misrecognition types in Sect. 2. According to the table, among the three types of misrecognition, "cross constraint" is the best (73.8% on core test set and 74.0% on complete test set). On core test set, the accuracy from "cross constraint" misrecognition is still lower than phone recognition which is trained by cheat phonetic question set (74.4%). However, the accuracy of the models trained by cheat question set drops when they are tested with complete test set. This is because cheat phonetic question set is adjusted only for core test set. In contrast, the accuracies of the models trained by "cross constraint" misrecognition are the same on both core and complete test set. This means that the quality of phonetic questions generated from the system is good enough and it is less susceptible to the change of test set or corpus than the manual phonetic questions.

For "count-limited", since the accuracy when the threshold is one is worse than zero ("direct" in Table 2), no more tests are performed for a higher threshold. Also, because the results of "direct" and "1-limited" are worse than "cross constraint", the tests are performed only on core test set.

In this phase, the accuracy of the models trained from the cheat question set is still better than the accuracy of the models trained from misrecognition. With the analyzing the number of questions, the number of questions from misrecognitions are

Table 3. Strategies to improve model quality for misrecognition

Type	Complete	# questions
backing-off	74.0	1,264
second	74.0	1,640

higher than the number of cheat questions. This means that in questions generated from misrecognitions, there are a lot of out-of-class errors. We hypothesize that if the models for misrecognition are better, the number of out-of-class errors should be reduced. In the next phase, we will improve the system accuracy by using better models for misrecognition.

5.2 Phase Two

Two strategies for increasing model quality are proposed. The first strategy is to use backing-off context-dependent models. Backing-off is very simple context-dependent generalization technique. This technique requires no phonetic questions for context-dependent generalization. When insufficient data for training a model exists, that model backs-off and some less informative but trainable model is used instead. For example, if a triphone has only a few examples in the training data, a biphone should be used. If a biphone is still not trainable, a monophone should be used. With this strategy, it is possible to insure that all models are well trained. The disadvantage of this strategy, however, is that the difference between more and less informative models is too large when a backing-off occurs.

From the above reason, the models generated from backing-off technique are worse than the ones generated from tree-based state tying. However, the misrecognitions generated from backing-off context-dependent models are better than the ones generated from context-independent models.

In this phase, only test on complete test set is shown. The accuracy of the models trained from “backing-off” increases up to 74.0% which is equal to the accuracy of the models trained from cheat questions. This indicates that better models can generate better misrecognition. Moreover, the quality of the questions generated from misrecognition are as good as cheat manual questions.

We also want to know that if we use the models trained above to generate misrecognitions again, use these misrecognitions to generate questions and train the models again from regenerated questions, is the result better? In Table 3, “second” shows the accuracy from these models. The accuracy is the same as “back-off”. This means that the models are saturated and no more improvement can be obtained. So there is no need to repeat these steps again. Moreover, “back-off” is slightly better than “second” since the number of questions from “second” is higher than the number of questions from “back-off”.

6 Conclusion

An alternative way to automatically generate phonetic questions has been presented. This technique employs misrecognitions to generate classes where each class is assumed to have similar properties. Then phonetic questions are generated from these class combinations. The quality of these questions is proved by the recognition result. The accuracy of the models trained from these questions is as good as the accuracy of the model trained from cheat questions. These questions are, however, more consistent than handmade questions which rely on judgment of human experts (for example, for the same TIMIT corpus, there are disagreements in linguistic classes between [9] and [10]). These questions are also more easily implemented in any new language without language specific linguistic knowledge since misrecognition-based question generation is an automatic process.

Acknowledgements

This material is based upon works supported by the Science Foundation Ireland for the support under Grant No. 02/IN1/I100. The opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Science Foundation Ireland.

References

1. Odell, J.J.: The Use of Context in Large Vocabulary Speech Recognition. Ph.D. Thesis. Cambridge University, Cambridge (1995)
2. Beulen K., Ney H.: Automatic Question Generation for Decision Tree Based State Tying. In Proc. ICASSP, Vol. 2 (1988) 805-809
3. Singh, R., Raj, B., Stern, R. M.: Automatic Clustering and Generation of Contextual Questions for Tied States in Hidden Markov Models. In Proc. ICSLP, Vol. 1 (1999) 117-1202
4. Willett, D., Neukirchen, C., Rottland, J. and Rigoll, G.: Refining Tree-Based Clustering by Means of Formal Concept Analysis, Balanced Decision Trees and Automatically Generated Model-Sets. In Proc. ICASSP, Vol. 2 (1999) 565-568
5. http://htk.eng.cam.ac.uk/
6. Tarsaku, P. and Kanokphara, S.: A Study of HMM-Based Automatic Segmentations for Thai Continuous Speech Recognition System. In Proc. the Symposium on Natural Language Processing, (2002) 217-220
7. Garofolo, J.S., Lamel, L.F., Fisher, W.M., Fiscus, J.G., Pallett, D.S., Dahlgren, N.L.: DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM, NIST (1993)
8. Lee, K.F. and Hon, H.W.: Speaker-Independent Phone Recognition Using Hidden Markov Models. IEEE Trans. Acoust., Speech, Signal Processing, 37(11) (1989) 1641-1648
9 Kanokphara, S. and Carson-Berndsen, J.: Feature-Table-Based Automatic Question Generation for Tree-Based State Tying: A Practical Implementation. In Proc. Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, (2004)
10 Chang, S., Greenberg, S. and Wester, M.: An Elitist Approach to Articulatory-Acoustic Feature Classification. In Proc. Eurospeech, (2001) 1725-1728