DESIGN OF A COMMAND INTERFACE WITH A DYNAMIC GRAMMAR SPEECH RECOGNITION ENGINE

S. Kruger, S. Chennoukh, J. Flanagan, L. Rothkrantz
Delft University of Technology–ITS
Zuidplantsoen 4, 2628 BZ Delft, The Netherlands
email: L.J.M.Rothkrantz@cs.tudelft.nl

ABSTRACT

This paper presents the design of an open grammar speech recognition engine. The engine is built to operate in real-time and kept simple. The objective of this paper is to show how task dependent knowledge can improve speech recognition accuracy by cancelling out parts of the command set depending on the state of the interface.

The system is a phone-based recognizer and therefore the vocabulary and grammar can easily be extended without retraining the system.

The hypothesis of this paper is that speech recognition accuracy will improve if higher level knowledge about the state of a task is used by the recognition engine.

1 Introduction

Recently, the first commercial automatic speech recognition systems (ASR) became available. The speech recognition technology is succesfully being applied in telephone inquiry systems. The keyboard of computers can be replaced by the human voice, to deliver short commands.

In all these applications, the grammar and the dictionary are limited. To design such a system from scratch takes a lot of time and effort in training acoustic models to a specific application. In another approach, a general ASR system, that theoratically can recognize any vocabulary, is restricted to a strictly limited number of words and a grammar. The general system contains pretrained models for speech units which can be combined to form words that are specific for a certein task.

To realize a real-time ASR system, a reduction of resources is necessary. The advantage of a pretrained general ASR system is that less effort is needed to apply it to a specific task. However, to realize a high recognition rate, additional training for the specific task is necessary.

Usually, a task can be split into a number of subtasks, and accordingly a subset of all possible commands. Depending on the state of the system, not all tasks can be performed, and not all commands are available at all times. For each subtask a separate ASR system could be built focussed on its commands, but this is a waste of resources. It is better to have one ASR system for a complete task, and have it activate parts of the command sets depending on the state of the system.

In this paper, we use the general approach to realize an ASR application. We train the acoustic models for general purpose, then perform additional training for a specific application. Then the impact of use of knowledge about the state of the system is examined.

2 Open Vocabulary

The idea to use one speech recognition system for several applications is not new. Ideally, a system that recognized a very large vocabulary can be used for many tasks. The concept to keep these large systems in proportion was to chop speech up into speech units, train the speech units, and during recognition, patch the units back together to form sensible language.

Many speech units have been proposed, from compound syllables, syllables, phonemes to sub-phone units. Although there is still a lot of research into which unit best matches actual speech, the phone is most widely used for its size (and the associated number of different ones), intuitive use and for historical reasons (used in dictionaries for ages).

For the large vocabulary speech recognition system a dictionary is created with, say, 60000 of the most used words in natural language. Usually, this is sufficient to support most tasks.

The next step towards application specific speech recognition while retaining the all-purpose engine is the open vocabulary method. Any task doesn't need all these 60000 words, but only a subset. But while a dictionary is created fairly easily, the acoustic models take a lot of effort to create. All the speech units are trained for general use, and a task specific vocabulary is constructed for each application.

3 Dynamic grammar speech recognition

Dialog systems already use a dialog structure to interpret the output of a speechrecognizer in terms of the expected input from the user. Most dialog systems take

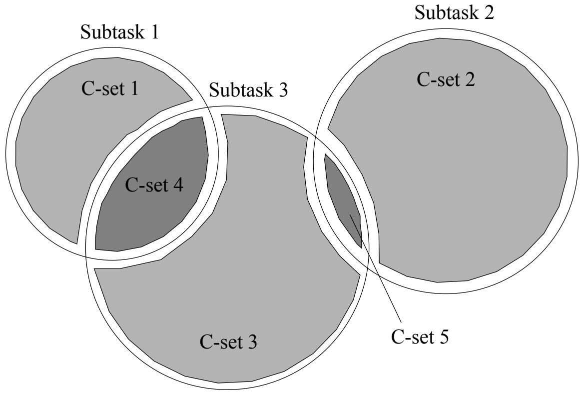

Figure 1: Commandsets corresponding with subtasks

input from a speech recognizer that recognizes natural language and then filter out the relevant words that the recognizer produces.

To do this, the dialog system uses a slot-filling algorithm. Depending on the state of the dialog, it expects the user to utter words from a small set that would be probable at the time. For example, the system may expect the user to utter a number and a product. It then creates a number and a product slot, and try to fill these slots with the input from the speech recognizer.

This knowledge about the state of the dialog can be used to the advantage of the speech recognizer, by telling it what to expect in the current situation. Not only accuracy, but also speed may improve because of the smaller set of options to consider.

In figure 1, the commands of three subtasks are denoted symbolically as Venn-diagrams. From these diagrams, 5 command sets are found. A speech recognition system can enable different sets when different subtasks are active. When subtask 1 is active, Command set 1 and 4 must be enabled.

A natural language speech recognizer has a language model based on statistical properties of natural language, such a the occurence probabilities of word-pairs or triples. To incorporate knowledge about the state of the dialog into such a language model requires to calculate conditional occurence probabilities. Calculating the general statistics about natural language is a huge task indeed and adding this knowledge would require even more effort.

This paper concentrates on speech recognition in command interface environments. Dialogs in such an environment may be more constrained in syntax. This allows to use a speech recognizer with a grammar language model rather than a natural language recognizer. The language model is not based on statistical properties of language, but is predefined for a task. The grammar specifies which are valid commands and what is their syntax and each command has equal occurence probability.

Based on the state of the dialog, the command set can vary. This knowledge can easily be integrated in the speech recognizer by enabling or disabling parts of the grammar. This is called a dynamic grammar speech recognition (DGSR). If the dialog system and the speech recognizer are integrated closely, they can share knowledge of higher level to optimize accuracy.

SIMPLE_COMMAND
CREATE
MOVE
DELETE
HIGHLIGHT
POP UP
POP DOWN

Figure 2: Word category

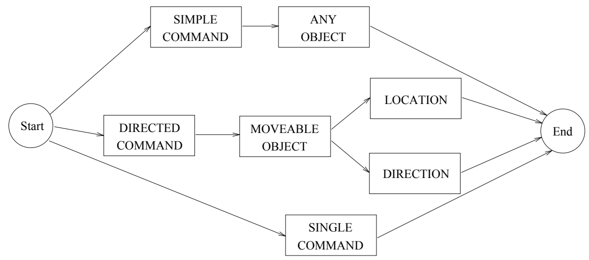

Figure 3: Grammar graph

4 Grammar

The grammar is structured in such a way that the speech recognition engine can easily find an optimal path through it when presented with input. A grammar compiler builds a graph from a grammar definition, in which the basic components are instances of the acoustic models.

A grammar is built up from word categories or word classes (see figure 2). One class category contains an arbitrary number of words or concatenated words, and works like an or-port. The grammar compiler constructs a parallel path for each word in the category. A category can be viewed as a node, because it has a number of arcs fanning in and a number of arcs fanning out, but the inside of the node is shielded.

A graph is constructed by defining the directed arcs between word categories (figure 3). The grammar compiler replaces each word by its phone sequence and each phone by an instance of its acoustic model plus some variables for kepping track of the recognition process.

The recognition process is really a scheme of updating scores of state instances in the graph and propagating these values from the start to the end node. At the end of a sentence, the optimal word sequence is traced back through the graph to present the resulting sentence.

5 Implementation

To test the goals of this paper, a speech recognition engine is used that can accept any grammar and recognize only sentences that are valid according to that grammar. The engine is an open-vocabulary, phone based speech recognizer based on the Hidden Markov Model method. The grammar is represented by a directed graph that connects word categories. Each word category may contain any number of words or word sequences. The word category works as an or-construct. A word category is instantiated in a node in the grammar graph. The arcs of the graph are specified by identifying the predecessors of each node. Every graph contains at least a start and end node.

Each path in the graph represents the syntax of a command, where each node may be substituted by one of its words. The speech recognition engine compiles a network from this graph by substituting each word by it's phonetic transcription and each phone by an instance of the acoustic model.

A Viterbi search finds the best path in this network, given a stream of features extracted from the input utterance. A evaluation algorithm determines the number of word errors. The speech recognizer can only produce sentences that are valid according to the grammar. Therefore, an error is at least a sentence that is in the grammar, and substitution errors are the most common. For a deletion or an insertion error to occur, the wrong sentence must be in the grammar.

It is easy to merge two grammars into one new one. All nodes of both grammars are copied into one file, except the start and end node are copied once and the end node's predecessor list is a concatenation of the two original ones. It is also easy to disable part of a grammar. If one arc in a path from start to end node is removed, the path becomes invalid and the recognizer will no longer recognize the corresponding syntax. It is of course wise to cut the path in the beginning of the network to reduce computational load.

6 Experiment

The objective of the experiment is to find how the recognition accuracy improves if a grammar is reduced to subtasks. To this goal, a grammar is designed that consists of 3 subgrammars, belonging to 3 subtasks.

The global task is the interface of a computer. The subtasks are Window Management (Finder), accessing Extensions and activating Control panels.

At some point in time, perhaps not all tasks are active. This means that the system need not expect the user to input any commands that belong to other tasks. Then that subgrammar may be disabled.

The testset from one subtask is first tested with the recognizer with a full grammar. It is interesting to see how the performance increases when subgrammars are disabled.

The speech recognizer is provided with simple context-independent acoustic models, monophones. The phones are represented by 3-state continuous density HMM's with a mixture of 4 Gaussians to represent the pdf of each state. The acoustic models are first trained on the TIMIT database. Initial models are estimated with a segmentation of the time-aligned TIMIT data. The embedded Baum-Welch algorithm is used to reestimate the models. The TIMIT database consists of clean speech of very good quality and low noise levels. To make the speech recognizer perform well in the testing environment, the acoustic models are retrained with speech recorded in the testing environment.

Because the system has context-independent acoustic models, not so much training data is needed to estimate the models. To make the system perform well for this experiment, the models are retreined on speech data from the grammar of the global task. This makes the context-independent models tend to model the phones in the context in which they appear in the task.

For the training and test data, 118 sentences were generated from the grammar. This set was divided in 3 subsets of 40, 40 and 38 sentences. Of 36 speakers, one of these subsets was recorded. 9 speakers are in the test set and 27 speakers are in the training set.

Of these 118 sentences, 51 are allocated to the task FINDER, 33 are allocated to CONTROL PANELS and 34 to EXTENSIONS. For the purpose of the experiment, four grammars are defined. One for each task and one general grammar, containing all three tasks.

In the experiment, first the complete test set is run through the general grammar. Then, the test sentences belonging to each task are run separately to see if any task performs better than another one. This is not interesting in itself, but for comparing the results. Next, the test sentences belonging to each task are run through the speech recognizer given only the task-specific grammar. The results of this experiment are combined to see how the dynamic grammar works on average for the complete test set.

To determine the accuracy of the system, and compare the different results, a metric is used. For the purpose of this paper, the word error rate is computed as follows:

$$\text{word error rate} = \frac{\text{word errors}}{\text{total number of words}} \cdot 100\%$$

$$\text{word errors} = \text{substitutions} + \text{deletions} + \text{insertions}$$

Here, a substitution, deletion or insertion all score 1 penalty. A match can be interpreted in different ways, for example, a substitution is also a deletion plus an insertion. The metric algorithm computes the optimum (minimum errors) error rate.

7 Results

The results of the experiment are summarized in table 1. In the left column, the grammar is written. General is

		TIMIT	Retrained
Grammar	Test		
General	FIND	54.7%	13.1%
Finder	FIND	46.9%	10.8%
General	CPAN	61.3%	4.6%
CPanels	CPAN	48.5%	4.6%
General	EXTS	43.7%	22.6%
Extensions	EXTS	36.8%	21.5%
General	ALL	52.8%	14.1%
Dynamic	ALL	44.1%	12.7%

Table 1: Word error rates

the combined grammar of the three subtasks Finder, CPanels and Extensions. The Dynamic grammar is the combined grammar, of which only the grammar of one subtask is enabled when this subtask is active. The next column contains the name of the test set. FIND is the set with commands for the FINDER, CPAN for the CONTROL PANELS and EXTS for the Extensions.
In the last two columns, the word error rates are printed for two different sets of acoustic models. The first is only trained on the TIMIT database. The second is reestimated with the training set recorded in the same environment as the test set.
Each test set is tested both on the general grammar and on the grammar specific for the subtask. As can be seen in the table, each time the speech recognizer does a bit better when it is given a smaller grammar.
The retrained models are significantly better. For the complete testset, the TIMIT trained models shows a 16.4grammar. For the retrained models, error decrease is 9.9models that are better at the acoustic match have less to gain from the dynamic grammar technique.

8 Conclusion

The results show that a dynamic grammar can improve the recognition rate significantly, although it works better for less perfect acoustic models. Still, the effort is worth the while, especially since robust systems tend to have more variable input and therefore less accurate acoustic models.
An issue that has not been investigated in this paper, but seems rather evident is that a dynamic grammar, by disabling parts of the grammar, reduces the amount of time needed to recognize speech. In a real-time application this may free up resources to improve accuracy or another quality-measure otherwise.

References

[1] J. Allan, Natural Language Understanding, Benjamin/Cummings, 1994.

[2] L. Ban, P. Tatai, Automatic Speech Segmentation for an Open Vocabulary Recognition System, Proc. ECSAP-97, 1997.

[3] D. Geller et al., Speech Recognition on Spheric, an IC for Command & Control Applications, EuroSpeech–97, 1997.

[4] J.L. Flanagan, I. Marsic, Isues in Measuring the Benifits of Multimodal Interfaces, Proc. ICASSP '97, 1997.

[5] Q. Lin, et al., Robust Distant Talking Speech Recognition, Proc. ICASSP 96, 1996.

[6] L.J.M. Rothkrantz, et al., The Simulation of a Human Operator by an ASP System, Proc. ASS97, 1997.

[7] V. Zue, Conversational Interfaces: Advances and Challanges, Proc. EuroSpeech–97, 1997.