SEARCHING WITH A TRANSCRIPTION GRAPH

Z. Li, P. Kenny and D. O'Shaughnessy
INRS-Télécommunications, Université du Québec
16 Place du Commerce, Verdun, Québec, Canada H3E 1H6

ABSTRACT

We present new developments in our approach to the search problem in very large vocabulary speech recognition. A transcription graph, which encodes all high scoring phonetic transcriptions which satisfy lexical constraints, is built so as to completely separate the search of the lexicon from the construction of the word graph. This enables us to limit the computational burden of searching in a way which is essentially independent of the size of the language model.

1. INTRODUCTION

We begin with a brief summary of our approach to the search problem [1]. We use a two-pass search strategy, which has the advantage that we can use inexpensive models in the first pass to radically prune the search space, and then scrutinize the data more closely using a powerful language model and detailed acoustic-phonetic models in the second pass. The pruned search space is represented by a word graph [1, 2]. The major bottleneck in the first pass is the search of the lexicon. We avoid the expense of a Viterbi search of the lexicon by drawing up a table of estimates of phone scores and durations, which we obtain by a backward Viterbi search of a much smaller graph which imposes diphone rather than full lexical constraints on phonetic transcriptions [3, 4]. These estimates of phone scores and durations can be used to calculate an approximate acoustic match for an arbitrary phonetic transcription at a cost of a single floating point operation per phone. In [1] we showed how this could be combined with the monotone graph search algorithm [5] to build a word graph in a first forward pass.

We have encountered two problems in extending these techniques to very large vocabulary searches. Firstly, the scheme of carrying out a stack search of the dictionary every time a word boundary is hypothesized requires devoting an unreasonably large amount of computation to stack management. Secondly the monotone graph search presented in [1] suffers from the disadvantage that searches of the dictionary with different initial conditions are carried out completely independently of each other.

Our new approach deals with both of these problems by building a new data structure which we refer to as a *transcription graph*, as an intermediate step between the backward Viterbi search and the monotone graph search (see Fig. 1). The idea is that the transcription graph encodes all possible phonetic transcriptions of the data which respect lexical constraints and have high acoustic phonetic scores. By traversing the transcription graph, we may build a word graph without any language model. Then we use the language model to construct a new word graph

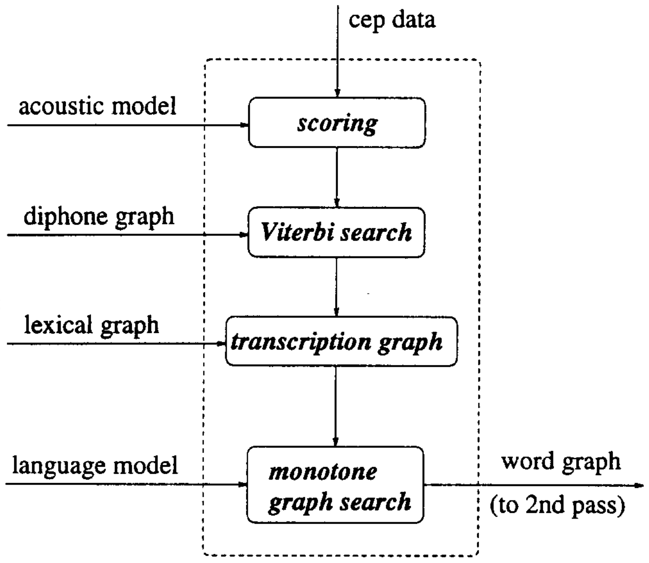

Fig. 1 The first pass search

using the monotone graph search algorithm. Accordingly, we do not have to carry out a search of the dictionary every time a word boundary is hypothesized (as in [1]).

2. THE TRANSCRIPTION GRAPH

Before giving a detailed description to the transcription graph, let us take a closer look at the lexical tree organization [6]. The lexical tree can be constructed such that every node has a phone label associated with it. Every complete path which starts from the root node and ends with a leaf node gives a phonetic transcription of a word. To visit the next word after the leaf node, we can go back to the root node of the lexical tree. Alternatively we can pre-compile the cross-word transitions into a structure which we call a *lexical graph.* The advantage of the latter approach is that it is useful for dealing with cross-word phonology; in the absence of cross-word phonology, the lexical graph can be constructed from the lexical tree by adding branches from each of the leaf nodes to all nodes which are at the first level of the lexical tree. There is a one-to-one correspondence between paths through the lexical graph and phonetic transcriptions which respect lexical and phonological constraints.

As for the transcription graph, it is defined in such a way that there is a one-to-one correspondence between paths through the transcription graph and the phonetic transcriptions *together with* their segmentations, where phonetic transcriptions are segmented by means of the table of phone scores and durations whose construction relies on the one-phone look-ahead property [3, 4]. We specify nodes in the transcription graph by means of triples (t, F, n) where n is a node in the lexical graph and F is a look-ahead phone hypothesized to start at time $t+1$.

The transcription graph can be constructed by a forward beam search as follows. Fix a time t. For each beam entry (t, F, n) and for each possible successor node n' in the lexical graph, advance the beam entry by one phone so as to obtain a new look-ahead phone F' and a new segmentation time t'. For each triple (t', F', n') created in this way, check to see if it is already on the beam at time t' and, if it is, update its forward score if a higher forward score has been found. We can assign a forward-backward score to a node (t, F, n) by combining its forward score with the backward score $\beta_t^*(F)$ [3, 4].

At each time t the beam entries can be pruned using a threshold against the highest forward-backward score of all nodes at that time. In deciding whether to prune a node (t, F, n), we take into account the position of n in the lexical graph: if a word boundary is reached we keep it, even though its score is low.

3. THE WORD GRAPH WITHOUT A LANGUAGE MODEL

After constructing the transcription graph, we may extract a word graph by a backward pass. Since the procedure so far only involves acoustic-phonetic scores, we refer to the resulting word graph as a *word graph without a language model.* A node in such a word graph is specified by a pair (t, F) where t, F have the same meaning as in the transcription graph. A branch joining a node (t, F) to another node (t', F') is labelled by a pair $(w, \mathbf{F})$. Here $\mathbf{F}$ is a transcription of a word w whose last phone is F'.

By exhaustively traversing the transcription graph in the reverse time direction, we create a condensed version of it, which contains only word boundary nodes. To fill in the information of a branch $(w, \mathbf{F})$ in the word graph, we may read the corresponding path in the transcription graph, which records acoustic-phonetic scores as well as segmentations of the phone string $\mathbf{F}$. For each node there is a backward score associated with it. This backward score is the acoustic score of the phonetic transcription which best matches all of the data in the future and which respects lexical constraints.

4. THE WORD GRAPH WITH A LANGUAGE MODEL

Given a word graph without a language model, we may use any type of finite state machine language model to create a new word graph. A node in this new word graph is labelled by a triple (t, F, σ), where t, F have the same meaning as the transcription graph, and σ is a state in the language model. A branch $(w, \mathbf{F})$ joining a node (t, F, σ) to (t', F', σ') is created only if there is a positive probability of taking the language model transition $\sigma \rightarrow \sigma'$ and generating a word w.

By the construction of the word graph without a language model, we know that the backward scoring function of the nodes satisfies the conditions of admissibility and monotonicity explained in [1]. Accordingly we can search this word graph to build the new word graph with a language model using the monotone graph search algorithm. (Recall that the principal operation in the monotone graph search consists of expanding a node. Each node n in the new word graph corresponds to a node m in the old word graph. Expanding the node n consists of simply reading all branches in the old word graph that originate in m.)

5. CONCLUSION

The lexical graph is ignorant of language model states and homophone distinctions. Our approach enables us to postpone the application of these high-level knowledge sources to a post-processor, and to carry out the acoustic-phonetic matching by searching a much smaller graph than would otherwise be necessary. In the absence of the high-level knowledge sources, pruning thresholds must be set conservatively. This is not a problem in our approach since our search is performed by table look-ups (but it could well be a problem for a classical Viterbi search). Relegating the language model to a post-processor ensures that the computational complexity of our algorithm is essentially independent of the size of the language model. (It is not clear how this could be achieved in a conventional Viterbi search with a back-off language model.)

ACKNOWLEDGEMENTS

This work was supported by the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[1] P. Kenny, P. Labute, Z. Li and D. O'Shaughnessy, "New graph search techniques for speech recognition," *Proceedings ICASSP 94*, vol. 1, pp. 553–556, April 1994.

[2] M. Oerder and H. Ney, "Word graphs: an efficient interface between continuous speech recognition and language understanding," *Proceedings ICASSP 93*, vol. II, pp. 119–122, April 1993.

[3] P. Kenny, P. Labute, Z. Li, R. Hollan, M. Lennig and D. O'Shaughnessy, "A very fast method for scoring phonetic transcriptions," *Proceedings Eurospeech 93*, vol. 3, pp. 2117–2120, September 1993.

[4] P. Kenny, P. Labute, Z. Li, R. Hollan, M. Lennig and D. O'Shaughnessy, "A new fast match for very large vocabulary continuous speech recognition," *Proceedings ICASSP 93*, vol. 2, pp. 656–659, April 1993.

[5] N. Nilsson, *Principles of Artificial Intelligence*, Tioga Publishing Company, 1980.

[6] P. Kenny, R. Hollan, V. Gupta, M. Lennig, P. Mermelstein and D. O'Shaughnessy, "A* - admissible heuristics for rapid lexical access," *IEEE Transactions on Speech and Audio Processing*, **1**(1), pp. 49–58, 1993.