A Decoder for Finite-State Structured Search Spaces

Diamantino Caseiro and Isabel Trancoso

Speech Processing Group
INESC/IST
Rua Alves Redol 9, 1000-029 Lisbon, Portugal
dcaseiro@speech.inesc.pt Isabel.Trancoso@inesc.pt

ABSTRACT

The theory of weighted finite state transducers (WFST) allows great flexibility in the early use of multiple sources of information in speech decoders. In this paper, we describe a decoder that relies on the algebra of WFSTs to integrate multiple sources of information in a one-pass search. The system has two modes of operation: time-synchronously for use with finite state problem specific grammars or with a word loop grammar for large vocabulary tasks; or time-asynchronously as a stack decoder also for large vocabulary recognition. Both modes of operation are decoupled from the language model. Experiments done with lattice rescoring tasks showed that the error rate is the same as with established state of the art decoders. Furthermore, experiments with explicit cross word pronunciation rules showed the feasibility of the inclusion of new knowledge sources early in the decoding process. We also found that the use of a time-synchronous search with a word loop grammar outperforms the stack decoder mode of operation by a factor of 10.

1. INTRODUCTION

The most widely used methodology to model the temporal characteristics of the speech signal in speech recognition is Hidden Markov Modelling (HMM). Although the theory behind that methodology is well understood and allows the use of simple algorithms to decode the utterance, actual systems for large vocabulary speech recognition are extremely complex, as they must deal with various sources of information at various levels, such as HMM states, phones, phones-in-context, cross-word triphones, pronunciation lexica, language models, etc. Most systems end up explicitly modelling every level of the decoder. This originates rigid, hard to modify systems in which new knowledge sources cannot be easily integrated without resorting to multiple passes.

The approach developed at AT&T Labs. [1] allows the early integration of knowledge sources by converting the problem to an algebraic form. The key to that approach is the notion of "weighted finite state transducer" [1], which extends the theory of "finite state transducers", commonly used in multiple areas of computer science, to transducers with weighted edges. They developed algorithms for their composition [2], determinisation and minimisation [3]. Most of the other systems that use finite state models use only acceptors, generally in the form of finite state grammars. The greatest advantage of transducers is the possibility of modelling not only the set of allowed sequences in each level, but also the relations between levels. With this approach, each component of the recogniser is modelled as a weighted transducer and the integration is performed by using the operation of transducer composition. The decoder itself can be very simplified.

The next section describes how to model the typical components of the recogniser as weighted finite state transducers. We continue by describing the proposed system, some preliminary recognition experiments, and directions for further research.

2. RECOGNITION SYSTEM COMPONENTS

In this section we will show how the typical components of a recogniser can be modelled as transducers.

Acoustic Models

Acoustic models implemented as HMMs can be compiled to WFSTs with a similar topology. The input label of each edge is a symbol representing the distribution. Its weight is the transition probability and the output label is epsilon, except for the edges leaving the initial state (or alternatively those arriving at final states) that output a symbol representing the model. Every path that traverses the transducer must output one, and only one, symbol. A transducer containing the set of all the models can be built using the concatenative closure of the union of the individual models.

Pronunciation Lexicon

We may regard a pronunciation as a transducer from phone sequences into a word. In most systems, a pronunciation is modelled as a phone sequence associated with a word (although some systems use more complex models [4]).

The pronunciation can be modelled as the trivial linear transducer with one edge for each phone in the sequence. Each edge has as input label the corresponding phone (or model), and has unitary weight. One of the edges in the sequence may have the pronunciation probability and must have the word as output; all other edges should have epsilon output. The linear transducer corresponding to the complete lexicon is the concatenative closure of the union of the individual pronunciation transducers. In large vocabulary continuous speech recognition, the lexicon is commonly organised as a tree [5]. We can transform the linear transducer to a

more general tree-like form, using, for example, the general transducer determinisation algorithm [3].

Language Models

Lattice or finite state language models may be trivially converted to WFSTs. Although n-gram language models may also be converted to WFSTs, exact conversions require a number of edges proportional to the number of all possible n-grams and not only to the number of n-grams observed in the training text. Hence, this conversion is only possible for small vocabulary and low order language models. The alternative is to resort to finite-state approximations. One approximation consists of creating a state for each context existing in the model. This state is the origin of all the edges that model n-gram probabilities with that context in the language model. Backoffs to lower order contexts are implemented as an epsilon edge from the higher order context to the lower. This is an approximation because there can be multiple paths in the automaton for the same n-gram with different probability values. From the point of view of the recognition, this can be a problem, because, if the backoff path probability is higher than the explicit probability, the former will be erroneously preferred. In practice, the approximation works pretty well being widely used in many disguises and variations [6] [7] [8].

Integration

Having implemented the previous components as transducers, the problem of decoding is reduced to searching for the best path in the transducer obtained with the composition of all the components.

In [9], the compositions included the levels from the acoustic model to the language model, resulting in a HMM-to-word WFST. A phone recogniser was used to decode the utterance. Later, in [10], the composition was extended to the level of the distribution, and the recogniser was tuned to deal with sequences of distributions (the self-loop was treated as a special case in the recogniser). During the composition of the various components, the operations of determinisation and minimisation were used to reduce the size of the transducer. The size of the resulting transducer was not much larger than the language model (in [10] it was respectively 2.1 and 2.5 times bigger for bigram and trigram language models) and allowed the efficient use of cross word acoustic models (the use of triphones only increased the size of the transducer in 2.5 %).

3. PROPOSED SYSTEM

The proposed system basically follows the same approach of implementing the various components as transducers, but we use different components, and a significantly different decoder.

In terms of acoustic modelling, we adopted a hybrid model that combines the temporal modelling capabilities of HMMs with the pattern classification capabilities of multilayer perceptrons (MLPs).

The system can operate as a stack decoder or time-synchronously. In both modes of operation, the language model is decoupled from the search, determinisation of the composition of the language model with the other WFSTs requires huge computational resources. This separation has the disadvantage that the exact language model probability cannot be included as early as in the AT&T approach. We plan to mitigate this problem in the future by using the unigram probability as an early approximation. Our approach does not require the use of approximations in the language model and allows the use of disk based language models to reduce the memory requirements [11].

The decoder has the following main inputs:

- The output of the neural network.
- A prior probability vector, to convert the probabilities estimated by the neural network to scaled likelihoods.
- An n-gram language model.
- A distribution-to-word WFST that is built outside of the system and that is usually the composition of the acoustic models (their HMM topology), the lexicon and a word loop or lattice.

Time-Synchronous Search

The time-synchronous search is used for lattice rescoring, and operates by the propagation of hypsets.

We define a hypset h_s^t as a set composed of hypotheses that end in the same state s of the WFST and at the same time t.

A hypothesis is a sequence of words with an associated cost, that is the combination of the best match of that sequence with the utterance and the cost (-LogProb) of the sequence in the language model.

We associate a hypset with each active state of the network. Initially, a hypset containing only the empty hypothesis is placed in the initial state of the WFST. Then the search starts: at each time instant, the hypsets are propagated through non-epsilon edges and merged at the destination state; then, the epsilon edges are traversed and the hypsets merged.

The merging of hypsets can be a relative time consuming operation. We try to reduce the impact of that operation by optimising some frequent cases: in general the network has a significant percentage of linear paths (sequences of states with only one successor, or with only a self-loop and a successor); and most of the output labels are epsilons.

We do some bookkeeping and share hypsets between states. Whenever possible, the hypset is dealt with as an opaque token that is shared and propagated from state to state, the differences in acoustic costs being kept separately at the states. Only when two or more different hypsets converge in a state, or when one hypset traverses an edge with a non-epsilon output label, are the hypsets really merged, and the decoder has to analyse each individual hypothesis. This bookkeeping is specially suited for the common case of "tree organised lexica" or the more general case of determinised WFSTs.

The language model score is applied to the hypotheses of the hypset as soon as an edge with an output label is traversed. We adopt a cache to reuse hypsets that were extended from the same hypset with the same word. This speeds up the propagation when the source state of that edge stays active in successive time instants.

During the merging of hypsets, the Viterbi criterion is enforced, and only the best of similar hypotheses is kept. When using n-gram language models, two hypotheses are considered similar if they have the same $n-1$ last words.

Stack Decoding

In the stack decoding mode the search proceeds at two levels: an outer word level which searches over the sequences of words (hypotheses), and an inner level search that extends a set of hypotheses (hypset) with another word.

Word Level Search Underlying the search at the word level is an array H_t, indexed in time, that, for each time instant, keeps the set of hypsets that ended at that time.

In the beginning of the search, we build a hypset containing only the empty hypothesis and ending at the initial state of the WFST. This pair is placed at H_0 and the search starts.

This word-level search consists only of a loop where the hypsets that end at each successive time instant are expanded. Notice that only one expansion is performed in each time instant, and all hypotheses ending at the same time are expanded simultaneously.

The expansion of the hypsets consists of extending its hypotheses with another word. At the end of the utterance we select the solution as the best hypothesis in a hypset ending at a final state of the WFST. The word expansion or state level search is time-synchronous.

State Level Search The state level search makes use of the distributions-to-words WFST build using the automata algebra. Because the search progresses word by word, the final edge of each word in this WFST must be labelled with a special end of word (EOW) label.

At the beginning of the expansion, the hypsets in the current time instant (all $h_i^t \in H_t$) are placed at corresponding word initial states (that is, all states following EOW edges from the final state of the hypset). Then a time synchronous search is performed. During this search, when an EOW edge is traversed the hypset is not propagated to the destination state, but is instead merged with the corresponding hypset at H_t, to build a hypset that will be latter expanded at time t.

Pruning

The use of pruning is fundamental not only for the efficiency of the search, but in most large cases it becomes essential because of the computational resources are always limited. We use three types of pruning:

Beam Search When the decoder operates in a non time-synchronous mode, we cannot know the value of the best state at a given time instant. To perform beam pruning we keep an array, indexed in time, of best-so-far costs that are updated as the search progresses. Whenever the best cost in a hypset is over a given threshold (beam) of the best-so-far at that time, the state containing the hypset is deactivated. During the merging of hypsets, every hypothesis over the threshold is removed from the resulting hypset. When a hypothesis is found that has a lower cost than the best-to-far at that time, the value in the array is updated.

When in time-synchronous mode, an additional pass is made over the active states, at each time instant, to deactivate states outside the beam.

This kind of pruning acts as a fundamental termination condition for the state level search in the stack decoding mode, as it proceeds until all states are deactivated.

Hypset window The hypsets only keep the best m hypotheses.

Phone Deactivation Pruning Because of the hybrid approach, the decoder has access to the posterior probabilities that are estimated by the neural network, and so we can use phone deactivation pruning [12].

4. EXPERIMENTS

Some experiments were done to evaluate the performance of the decoder, using the same framework as the Audimus system [13], a hybrid LVCSR system developed for European Portuguese. We tried to use as many components of that system as possible, in order to allow a direct comparisation. The Audimus system is based on the Noway decoder [12], one of the fastest hybrid decoders reported.

Neural Network The input of the MLP consists of a window of PLP vectors (26 parameters: 12 coefficients + energy + deltas) extracted from the speech signal. Each window has 7 vectors: 3 vectors on the right and 3 on the left of the central vector. The 3-layer MLP has 1000 units in the hidden layer, and 39 output units, corresponding to 38 context independent phones plus silence.

Acoustic Models The acoustic models use the same architecture as used in Audimus: a sequence of states with no self-loops to enforce the minimal duration of the model, and one final state with a self-loop.

Pronunciation Lexicon The pronunciation lexicon has standard or canonical pronunciations for each of the 5000 words in the vocabulary. Most words have only one pronunciation although some have multi-pronunciations. A short-pause acoustic model was added at the end of each pronunciation to model eventual pauses between words.

Rule System In order to experiment with the expressive power of the WFST-based system, we developed a small pronunciation rule specification system dealing with cross-word pronunciation rules.

These pronunciation rules are specified using a finite-state grammar. The syntax of the rule specification language is similar to the BNF rules augmented with regular expressions. Each rule is represented by a regular expression augmented with the operator $\rightarrow$, simple transdution, such that (a $\rightarrow$ b) means that the terminal symbol a is transformed into the terminal symbol b. The language allows the definition of non-terminal symbols (e.g. $vowel), but they can only be used after being defined (this restriction disallows recursive rules and enforces the finite-state nature of the grammar).

All the rules are optional, and are compiled into WFSTs. The union of the individual rule transducers is then composed between the lexicon and the phone models. Hence, we have two-level rules that map phone sequences to canonical phone sequences (that is, those obtained from the concatenation of lexicon entries).

Word Lattice We use word lattices obtained from a previous run of the Noway decoder to test the decoder in acoustic rescore mode.

Language Model We use a trigram language model trained from 46 million words from the online edition of the PÚBLICO newspaper, corresponding to the years from 1995 to 1998. The language model has 5.3 million trigrams and 927 thousand bigrams and a test set perplexity of 107.

Lattice Decoding

The first set of experiments was done during development with the purpose of helping to detect problems and calibrate the synchronous search. In these experiments, lattices of word hypotheses generated by the Noway decoder were used as a constraint to the decoder and no look-ahead information was used. The lattices had an average of 136 word arcs per spoken word. We performed some experiments with the expansion of words to phones and found that the use of language model probabilities as soon as possible allows the use of tighter beams. The best performance was obtained by placing the arc with the word output at the beginning of the pronunciation and sharing the suffixes of the words arriving at the same state.

The lowest error rate obtained with a reasonable beam (50.0) was 16.3%. Some experiments were performed to find the best values for the language model scale (5.0) and word insertion penalty (0.0). This WER is similar to the best error rate obtained by the Noway decoder using the same models (16.5%).

Rule Decoding

We performed some tentative experiments using some unweighted cross-word pronunciation rules for European Portuguese. The rules were converted from the sandhi rules of the DIXI European Portuguese text-to-speech system [14]. The rules were applied in the same conditions as the lattice decoding experiments. No degradation was observed, but the improvements were very slight and not significant. The WFSTs obtained with the use of the rules were 3 times larger than the original ones, and the decoding was also 3 times slower.

Although the accuracy results were not convincing, these experiments served to demonstrate the flexibility of the system.

Large Vocabulary Tasks

We performed some experiments with large vocabulary recognition using either the stack decoder mode or a time-synchronous search based on a word loop grammar. We observed that the time-synchronous search is faster than the stack decoder by a factor of 10 times. In table 1 we present the results obtained with the word loop grammar. The results where obtained using a 600Mhz Pentium III PC runnig linux.

WER	xRT
25.9	1.0
19.2	1.7
17.2	3.0
17.0	4.2
16.8	6.9
16.3	10.1

Table 1: LVCSR Results using a word loop grammar.

5. CONCLUSIONS AND FUTURE WORK

We have presented a decoder that uses the algebra of weighted finite state transducers, to include knowledge sources early in a one-pass search. This system operates mainly time synchronously for use with finite state problem specific grammars and for large vocabulary tasks using a word loop grammar. The system can also operate as a stack decoder for large vocabulary recognition with wide context language models.

We are also extending the system to allow the use of gaussian mixtures distributions instead of the neural network. The system is currently able to read and write acoustic models trained with the HTK [15] system, and read acoustic vectors from the same system. It can also determine the probability of an acoustic vector in a gaussian mixture distribution. However, those capabilities are not yet integrated in the search algorithms. After this extension, we plan to test the system with both word-internal and cross-word triphones.

6. ACKNOWLEDGEMENTS

The authors would like to thank Prof. João Paulo Neto for providing the resources from the Audimus system that permitted the evaluation of the system, and for many fruitful discussions. The present work is part of Diamantino Caseiro's PhD thesis, initially sponsored by a FCT scholarship (PRAXIS XXI/BD/15836/98). The early part of this work was integrated in the REC project. The more recent developments were done in the framework of the ALERT project.

7. REFERENCES

1. F. Pereira, M. Riley, "Speech Recognition by Composition of Weighted Finite Automata" in Finite State Language Processing, E. Roche and Y. Schabes Eds, MIT Press 1997.

2. M. Mohri, F. Pereira, M. Riley, "Weighted Automata in Text and Speech Processing", Proceedings of the ECAI 96 Workshop, 1996.

3. M. Mohri, "Finite-State Transducers in Language and Speech Processing", Computational Linguistics, 23:2, 1997.

4. M. Adda-Decker, L. Lamel, "Pronunciation Variants Across Systems, Languages and Speaking Style", Proceedings of the Workshop Modelling Pronunciation Variation for Automatic Speech Recognition, Rolduc, 1998.

5. R. Haeb-Umbach, H. Ney, ”Improvements in Beam Search for 10000-Word Continuous-Speech Recognition”, IEEE Transactions on Speech and Audio Processing, vol.2, no. 2, April 1994.

6. G. Riccardi, E. Bocchieri, R. Pieraccini, ”Non Deterministic Stochastic Language Models for Speech Recognition”, Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP’95), 1995.

7. V. Valtchev, J. Odell, P. Woodland, S. Young, ”A Dynamic Network Decoder Design for Large Vocabulary Speech Recognition”, Proceedings of the 1994 International Conference on Spoken Language Processing (ICSLP’94), Yokohama, Japan, 1994.

8. F. Weng, A. Stolcke, A. Sankar, ”Efficient Lattice Representation and Generation”, Proceedings of the 1998 International Conference on Spoken Language Processing (ICSLP’98), Sydney, Australia, 1998.

9. M. Mohri, M.Riley, D. Hindle, A. Ljolje, F. Pereira, ”Full expansion of context-dependent networks in large vocabulary speech recognition”, Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP’98), Seattle, Washington, 1998.

10. M. Mohri, M. Riley, ”Integrated Context-Dependent Networks in Very Large Vocabulary Speech Recognition”, Proceedings of the 6th European Conference on Speech Communication and Technology (Eurospeech’99), Budapest, Hungary, 1999.

11. M. K. Ravishankar, ”Efficient Algorithms for Speech Recognition”, PhD thesis, School of Computer Science, Carnegie Mellow University, 1996.

12. S. Renals, M. Hochberg, ”Efficient search using posterior phone probability estimates”, Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP’95), Detroit, USA, 1995.

13. J. Neto, C. Martins, L. Almeida, ”A large vocabulary continuous speech recognition hybrid system for the Portuguese language”, Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP’98), Sydney, Australia, 1998.

14. L. Oliveira, M. Viana, I. Trancoso, ”A rule-based text-to-speech system for Portuguese”, Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP’92), San Francisco, California, 1992.

15. S. Young, J. Jansen, J. Odell, D. Ollason, P. Woodland, ”The HTK Book, version 2.1”. Distributed with the HTK toolkit.