Large Vocabulary Continuous Speech Recognition Using Weighted Finite-State Transducers

Diamantino Caseiro and Isabel Trancoso

[1] INESC-ID/IST, Rua Alves Redol 9, Lisbon, Portugal
{dcaseiro,Isabel.Trancoso}@l2f.inesc-id.pt

Abstract. Weighted finite-state transducers are an unifying formalism for the implementation and integration of the various knowledge sources and structures typical of a large vocabulary continuous speech recognition system.
In this work we show how those knowledge sources can be converted to this formalism, and how they can be integrated in an optimized network, using our finite-state library and tools.
Experiments performed using our system showed the importance of the optimization of the integrated network, and allowed us to obtain very significant improvements in the speed of the recognizer.

1 Introduction

A finite-state transducer (FST) is a computational model similar to a finite-state automaton, but with an extra label associated with each edge. That *output* label, allows the FST to model *relations* and mappings between languages, while retaining some of the excellent computational properties of finite-state automata. FSTs have been widely used in natural language processing, but their use in automatic speech recognition (ASR) is still not widespread, although there has been a growing interess in the community, specially following the pioneering work on the use of weighted finite-state transducers (WFSTs) developed at AT&T Labs.

In the WFST approach, the recognition problem is reduced to building an *integrated network* using WFST composition of all the components in the system, and then searching for the best path in the network.

Different systems use different components, but a typical system can include $S \circ A \circ C \circ P \circ L \circ G$, where S represents the acoustic speech signal, A the acoustic models, C the context-dependency transducer, P represents phonological rules, L the lexicon and G the language model.

In [1], the compositions included the levels from the acoustic model to the language model, resulting in a *phone_id-to-word* WFST ($C \circ L \circ G$). A phone recognizer, constrained with the network, was used to decode the utterance. Later, in [2], the composition was extended to the level of the distribution ($A \circ C \circ L \circ G$). As the resulting network N was very large and contained many long linear paths[2], it was factorized into two transducers $N = H \circ F$, where H replaces each linear path with a unique identification code, and F is similar to N but with each linear path replaced with an edge labeled with the linear path code. The recognition was performed using the previous recognizer and the F network. The linear paths were directly supported as "phones" in the phone recognizer. During the composition of the various components, the operations of determinization and minimization were used to reduce the size of the intermediate transducers. The size of the resulting transducer was not much larger than the language model (in [2] 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 cross word triphones only increased the size of the transducer in 2.5%).

The Spoken Language Systems Group of the Massachusetts Institute of Technology (MIT) has been also developing a similar approach, in their Jupiter conversational system. One of the main characteristics of their work [3] is the use of phonological rules (represented by a WFST P) in the recognition cascade $C \circ P \circ L \circ G$.

In the next section we describe how to model the components of a state-of-the-art large vocabulary continuous speech recognizer (LVCSR) as weighted finite-state transducers. In section 3, we present our finite-state speech recognition system, and in section 4, some recognition experiments. Finally, in section 5, we summarize the main conclusions.

2 Representation of Speech Recognition Components as WFSTs

2.1 Acoustic Models

Acoustic models implemented as hidden Markov models (HMMs) can be compiled to WFSTs with a similar topology. The input label of each edge is a symbol representing the distribution of the destination state of the HMM. Its weight is the transition probability and the output label is epsilon, except for the edges leaving the initial state that output a symbol representing the model. Every path that traverses the transducer must output one, and only one, symbol. The transducer A containing all the models can be built using the concatenative closure of the union of the individual models A_i $A = (A_1 \cup A_2 \cup \ldots \cup A_n)^*$.

2.2 Context Dependency

In a context independent system, each acoustic model described in the previous section can represent a specific phone. But most state-of-the-art recognition systems are context dependent. That is, the acoustic model used to represent a phone is selected depending on the particular context of the phone (for example, in triphone systems each phone has different models for each combination of

[2] A linear path is a path where its states, other than the first and the last, have at most one incoming and one outgoing transition.

previous and next phones). Context dependent systems thus allow much more detailed modeling.

The transformation from context independent to context dependent can be modeled by interposing a transducer C between the acoustic models and the lexicon, that will implement the context dependent to independent relation[3][1].

To build the transducer C we build its inverse by connecting the edges as show on table 1 for each triphone[4] $a-b+c$, right biphone $b+c$, left biphone $a-b$ and uniphone b[5]. Those edges connect states that represent particular contexts.

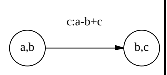
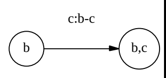
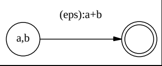
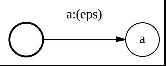
$\forall_{a-b+c}$ 	$\forall_{b-c}$ 
$\forall_{a+b}$ 	$\forall_{a}$ 

Table 1. Construction of the inverse context dependency transducer.

2.3 Pronunciation Lexicon

We may regard a pronunciation as a transducer from phone sequences into a word. In most systems, the pronunciation of a word is modeled as a sequence of phones.

The pronunciation can be modeled 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 transducer L corresponding to the complete lexicon is the concatenative closure of the union of the individual pronunciation transducers. In LVCSR systems, the lexicon L is commonly organized as a tree [4]. We can transform the linear transducer to a tree form by sharing prefixes, or we can transform if to an even more general form, using the generic weighted transducer determinization algorithm [5].

2.4 Language Models

There is a long tradition of using finite-state acceptors to model sentences in speech recognition. They are used in limited domains and in dialog systems to specify the acceptable sentences. In large vocabulary applications they are used

[3] That is, a transduction from the models that represent context dependent units to the lexicon that is represented in terms of context independent phones.
[4] $a-b+c$ should be read as the version of b that has a on the left and c on the right.
[5] We name a context independent unit *uniphone* when used in a system that also contains context dependent units.

to constrain the search in latter passes of multiple pass systems. Such lattice or finite-state language models can be trivially converted to WFSTs. However, the main language model methodology in LVCSR consists of local stochastic grammars specified as word n-grams.

One simple way to convert an n-gram language model to an automaton G, consists of creating a state for each possible context, and placing weighted edges between contexts labeled with the word and weighted with the n-gram probability. This exact conversion requires 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. Back-offs to lower order contexts are implemented as an epsilon edge from the higher order context to the lower. This is an approximation, because there may be multiple paths in the resulting 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 back-off path probability is higher than the explicit probability, the former will be erroneously preferred. In practice, the approximation works very well, being widely used. In figure 1 we show an example of a finite-state approximation to a trigram back-off language model with a vocabulary of two words a and b. Each state is labeled with the context it represents (the empty context is shown as $*$, the probability weights as P and the back-off weights as B).

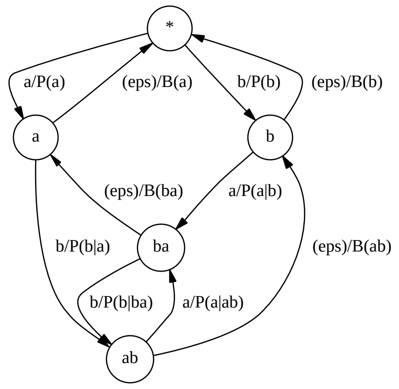


Fig. 1. Finite-state approximation to a trigram language model.

2.5 Optimization of the integrated network

One of the advantages of the WFST approach is that the integrated network resulting from the composition of the various component WFSTs can be optimized. The optimization is performed by combining 4 main operations:

Epsilon removal can be performed in some of the WFSTs. But it has to be done with care, as its use can blow up the size of the transducer (it should not be done on the approximated n-gram language model, for example).

Determinization is very important, as it reduces the redundancy of the integrated network, allowing prefix sharing.

Pushing redistributes the weights of the network without changing its topology. Pushing the weights towards the initial state has a very positive effect on the search. It can be seen as implementing a generalization of language model "look-ahead". Pushing is performed as part of the weighted minimization algorithm.

Minimization can be performed after determinization and returns the minimal deterministic equivalent transducer. Its main effect is the reduction of the memory required to use the network.

Determinization, pushing and minimization should be performed on the integrated network, but these operations can also be applied at various stages during the composition of the components WFSTs in order to reduce the size of the intermediate results.

Composition of the lexicon with the language model The integration of the speech recognition components using WFST composition, although conceptually simple, is a difficult task because of the size of the transducers involved. In particular, the determinization of the composition of the lexicon with the language model requires very large amounts of memory due to the very large size of the language model WFST.

In the AT&T approach, the lexicon transducer is linear [6] and it is disambiguated by adding dummy symbols to the end of the pronunciations. The lexicon can be ambiguous due to homophone words, or due to the pronunciation of some words being a prefix of other words. It is very important to disambiguate the lexicon, otherwise the determinization algorithm might not terminate. Another important aspect is that the output label should be the produced by the first edge of the pronunciation (to avoid the generation of non-coaccessible states[7] during composition). The linear lexicon is then composed with the language model, and the resulting transducer is determinized using the general weighted transducer determinization algorithm. This final determinization step requires an order of magnitude more memory than the size of the resulting transducer[8].

To address the problem of the memory required to build and optimize the integrated network, we developed a specialized algorithm for the composition of the lexicon with the language model [6]. That algorithm performs composition, determinization and pushing in one step, allowing an "on-the-fly" implementation [7]. It is also very memory efficient, requiring approximately the same

[6] Meaning that no part of the pronunciations is shared.

[7] Non-coaccessible states have no path to a final state.

[8] The main reason is that the determinization algorithm for weighted transducers is similar to the classic subset construction algorithm and needs to store a *set* of original states for each state of the result WFST.

memory as the resulting transducer. It can approximate the minimization step, yielding a transducer only 2-5% larger than the minimal one. The algorithm is based on the fact that the composition of deterministic (sequential) transducers is also deterministic[8]. The usual composition algorithm cannot be used to exploit this result, as it would generate impractical amounts of non-coaccessible states. Our algorithm avoids the generation of those states by using look-ahead information on the lexicon transducer.

3 Finite-State Recognition System

3.1 Finite-State Library

In order to build a speech recognition system we started by implementing an object-oriented library to manipulate finite-state machines. The design of the library closely followed the one presented in [9]. The library allows the representation of finite-state machines and their manipulation. Most finite-state operations are represented as sub-classes of an abstract finite-state machine, and have "on-the-fly" or "lazy" implementations, so that very complex expressions can be represented and used, without the need for very large intermediate results.

The library was designed from the start to be used in speech processing and allows the efficient use of very large automata with up to tens of millions of states. It also includes specialized modules for manipulating acoustic models, lexicon models, language models, acoustic vectors and other data used in speech processing. One particularity is the use of specialized classes that have the interface of finite-state machines, but that are implemented using the Turing-machine power of a computer language. For example, the context-dependency transducer shown in section 2.2 requires a number of states that is the square of the size of the alphabet, making it impractical for some frequent tasks with many thousands of symbols in the alphabet[9]. The library includes a WFST class that implements this transducer using very little memory.

3.2 Command Line Tools

The library was used to develop command line tools to convert automatic speech recognition components in standard file formats to WFSTs. Other tools implement WFST operations such as: composition, determinization, label pushing, weight pushing, our algorithm for the composition of the lexicon with the language model, etc

3.3 Decoder

The speech recognizer or decoder is based on a time-synchronous token-passing implementation of the Viterbi algorithm. Its main characteristic is that it has

[9] For example, to build segment pairs in speech corpora for use in concatenative speech-synthesis.

very little knowledge of the structure of the search space. As it is specified as a *distributions-to-words* WFST, previously build using the finite-state command line tools. The decoder can accept either Gaussian-mixture distributions or scaled likelihoods obtained from the output of a neural network (hybrid mode). It can be used to find the best hypothesis or it can generate a lattice or word graph.

One version of the decoder accepts as input the acoustic A, lexicon L and language model G WFSTs, and performs the composition and optimization of $A \circ L \circ G$ dynamically "on-the-fly", using our algorithm. The overhead of building the network "on-the-fly" is only about 20% in the latest implementation.

4 Recognition Experiments

In this section we describe some recognition experiments performed to evaluate the WFST approach. All the experiments were based on the BD-PÚBLICO corpus[10] (a European Portuguese corpus equivalent in size and purpose to WSJ0). The experiments were performed using a standard 600MHz pentium III PC, with 1GB of RAM.

We used a European Portuguese lexicon with 27k words and trigram backoff language models, trained from 46 million words from the online edition of the PÚBLICO newspaper, corresponding to the years from 1995 to 1998. The lexicon and language models were converted to WFSTs.

The acoustic model topology consisted of a sequence of states with no self-loops to enforce the minimal duration of the model, and one final state with a self-loop. The acoustic models were encoded into a single acoustic model WFST.

The acoustic observation distributions were modeled using a combination of the output of various neural networks[11].

In order to analyze the effects of the various finite-state optimizations on the integrated network, we plotted the variation of the word error rate (WER) and recognition time when using different pruning factors.

In figure 2, we show the performance obtained when using three integrated composition networks with various degrees of optimization. The first one was determinized ($det(A \circ L \circ G)$); the second one was determinized and pushed ($push\ det(A \circ L \circ G)$); the third one used a minimal language model ($push\ det(A \circ L \circ (min\ push\ det(G)))$); and finally we used a minimal integrated network ($min\ push\ det(A \circ L \circ G)$);

We see that, as expected, the network optimizations did not change the asymptotic WER. But they led to a dramatic improvement of the speed of the recognition system.

5 Conclusions

In this paper, we presented finite-state methods that allow the various knowledge sources and structures of a large vocabulary continuous speech recognition

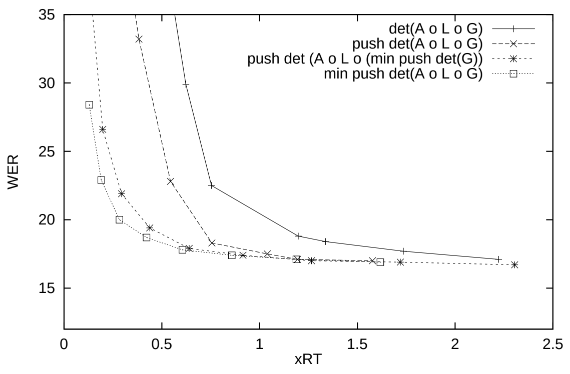


Fig. 2. Effect of various optimizations of the integrated network.

system to be implemented in the unifying formalism of WFSTs. In particular, we described our finite-state library and how it was used to implement a toolbox that allows the conversion of standard speech recognition components to WFSTs. Our recognition experiments verified the importance of the optimization of the integrated network, almost reaching the asymptotic accuracy in real time.

6 Acknowledgements

The present work is part of Diamantino Caseiro's PhD thesis, initially sponsored by a FCT scholarship (PRAXIS XXI/BD/15836/98). This work was also partially funded by IST-HLT European program project ALERT. INESC-ID Lisboa had support from the POSI program of the 'Quadro Comunitario de Apoio III'.

References

1. M. Mohri, M. Riley, D. Hindle, A. Ljolje, and F. Pereira. Full expansion of context-dependent networks in large vocabulary speech recognition. In *Proc. ICASSP '98*, Seattle, USA, May 1998.
2. M. Mohri and M. Riley. Integrated context-dependent networks in very large vocabulary speech recognition. In *Proc. Eurospeech '99*, Budapest, Hungary, September 1999.
3. J. Glass, T. Hazen, and I. Hetherington. Real-time telephone-based speech recognition in the jupiter domain. In *Proc. ICASSP '2001*, Utah, USA, May 2001.
4. R. Haeb-Umbach and H. Ney. Improvements in beam search for 10000-word continuous-speech recognition. In *IEEE Transactions on Speech and Audio Processing*, April 1994.
5. M. Mohri, F. Pereira, and M. Riley. Weighted automata in text and speech processing. In *ECAI 96 Workshop*, August 1996.
6. D. Caseiro and I. Trancoso. On integrating the lexicon with the language model. In *Proc. Eurospeech '2001*, September 2001.

7. D. Caseiro and I. Trancoso. Transducer composition for "on-the-fly" lexicon and language model integration. In *ASRU 2001 Workshop*, December 2001.
8. M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, June 1997.
9. M. Mohri, F.Pereira, and M. Riley. A rational design for a weighted finite-state transducer library. In *Automata Implementation. Second International Workshop on Implementing Automata, WIA '97.* Springer Verlag, 1998. Lecture Notes in Computer Science 1436.
10. J. Neto, C. Martins, H. Meinedo, and L. Almeida. The design of a large vocabulary speech corpus for portuguese. In *Proc. Eurospeech '97*, September 1997.
11. H. Meinedo and J. Neto. Combination of acoustic models in continuous speech recognition hybrid systems. In *Proc. ICSLP '2000*, Beijing, China, October 2000.