A COMPARISON OF PREFIX TREE AND FINITE-STATE TRANSDUCER SEARCH SPACE MODELINGS FOR LARGE-VOCABULARY SPEECH RECOGNITION

Hans J.G.A. Dolfing

Philips Research Laboratories
Weisshausstrasse 2
D-52066 Aachen, Germany
Email: Hans.Dolfing@philips.com

ABSTRACT

In this paper, we compare a large-vocabulary speech decoder, based on a phonetic prefix tree, with a decoder based on finite-state transducers. On the example of a large-vocabulary, isolated word recognition task without language model, we investigate the error-rates based on different beamwidths for fully optimized, tree-like and factored finite-state transducer networks. The results show that the decoder, based on a fully optimized finite-state network, achieves the same error-rates as the prefix tree decoder, using roughly 50% of active states. In addition, we modify the traditional phonetic prefix tree to a new tree, compatible with standard LVCSR decoders but optimized across all hidden Markov models to remove redundancies, which achieves better error-rates at small beamwidths.

1. INTRODUCTION

In the context of the weighted finite-state transducer (FST) approach to speech recognition, we compare the large-vocabulary continuous speech recognition (LVCSR) strategy as discussed in [1, 2] with a finite-state transducer decoder similar to [4, 5]. The comparison is on a large-vocabulary, isolated word recognition task of fullnames without language model because we concentrate on how to exploit the lexical model in the best way.

This study is motivated by the fact that there is much progress in speech recognition with finite-state transducer. For example, [6, 3, 4] all address aspects of how to improve LVCSR with finite-state transducers compared to the initial work in [5]. The mentioned papers all involve a time-synchronous Viterbi search, based on a precompiled finite-state transducer network.

Finite-state transducers have the principal advantage of concentrating the speech decoding into one compact, minimally sized, transducer network. Finite-state transducers have the potential of combining the early use of all available knowledge sources, within a well understood theoretical framework. In this way, we combine in a single decoding pass the advantage of a small, minimal finite-state transducer for speech decoding with the smallest beamwidth in decoding and best possible decoding results. However, it is not clear which finite-state transducer optimizations contribute most to the reduction of the number of active hypotheses in the Viterbi search.

For example, [5] shows the speedup of a LVCSR task in terms of real-time factors while applying more and more optimization steps like determinization, minimization and factorization to a precompiled finite-state transducer network. The speedup is due to the reduced number of active state hypotheses. However, the contribution of each step to the reduction of the number of active hypotheses is not clear.

In a traditional LVCSR decoder, we aim to minimize the number of active hypotheses in the time-synchronous beamsearch by applying all knowledge source as early as possible. This includes the application of language model lookahead (smearing), and acoustic (phoneme) lookahead.

Additionally, we realize that popular LVCSR decoders [2] are mostly based on Viterbi search based on a phonetic tree network. Loosely speaking, the main difference between the traditional phonetic tree decoding and decoding with finite-state transducer networks, is that the finite-state transducer networks also merges all word suffixes, i.e., the network is more compact.

To gain more insight into the effects of finite-state transducer optimization techniques on the size of the dynamic Viterbi search space, we compare in detail a LVCSR decoder [1] and a newly developed finite-state transducer decoder on a large-vocabulary, isolated word recognition task without language model. The aim is to quantify the effects, e.g., of suffix merging with minimization and factorization to merge several ‘phoneme arcs’ into one arc.

In Section 2 and 3, we discuss the general properties and optimizations of a recognition system based on finite-state transducers. After a short description of the experimental setup in Section 4, we report results in Section 5. Finally, we summarize the findings in Section 6.

2. FINITE-STATE TRANSDUCERS

Typically, within a finite-state transducer-based recognition system, the various constraints such as language model, lexicon, phonological rules, context-dependency, HMM topology, etc., are each represented as a possibly weighted trans-
ducer, and these transducers are composed together to form the single transducer to be used for recognition [5, 4].

In [4], we typically used the weighted finite-state transducer $CPLG = opt(C \circ P \circ L \circ G)$ for recognition, with $G$ is an $n$-gram language model, $L$ the lexicon, $P$ is a set of phonological rules, and $C$ adds context-dependent phonetic models. In the current paper, we typically use the transducer $TCL$ where $T$ maps the symbolic triphones to a triple of Gaussian mixture-ids. In addition, we use factoring [5] which replace several arcs with mixture-ids by one arc with a long chain of mixture-ids. Note that no language model is used because we concentrate on the effects of the search space modelings of the lexical information $L$. The optimization operator $opt(\cdot)$ performs $\epsilon$-removal, weight pushing, determination, and minimization. The operator $det(\cdot)$ performs determination only and $fac(\cdot)$ does factorization. Optimized transducer networks are denoted, e.g., as $optTCL$ while a (phonetic) tree-like transducer, which is deterministic by construction, is denoted as $treeTCL$.

3. SEARCH SPACE MODELING WITH FST

Traditional LVCSR, based on a phonetic tree, models the search space with a triphone prefix tree identical to the CL transducer in the left part of Figure 1. This example models two pronunciation variations of the word ‘THE’. The right side of Figure 1 shows $treeTCL$ which is the ‘internal’ representation where the triphones are generalized with CART to a triple of mixture-ids. In this paper, the input symbols with three mixture-ids are expanded on-the-fly by the decoder to a six state hidden Markov model where every mixture-id is expanded to two states.

![Fig. 1. The transducers treeCL (left) and treeTCL (right).](image)

In the rest of this paper, we refer to the transducers in Figure 1 as tree-like transducers and used the shorthand $treeTCL$. These transducers are deterministic by construction and are constructed with a special tool which takes a lexicon as input and produces $treeTCL$ as output.

It is obvious from $treeTCL$ in Figure 1 that the same mixture-ids are used for both pronunciation variations at the word start. To exploit this and make the transducer more compact, we have to enlarge it first with the composition $STCL = S \circ TCL$ which results in Figure 2 where the transducer $S$ maps sequences of mixture-ids (typically three) to one mixture-id per arc.

![Fig. 2. The transducer STCL.](image)

If we apply $treeFSTCL = fac(det(STCL))$ then we get the tree-like transducer on the left side of Figure 3. Note that this is still a tree just like the original phonetic tree $CL$. Only the labels on the edges have variable length and the common prefixes of mixture-ids are combined. In contrast, a full optimization $optFSTCL = fac(opt(STCL))$ gives the transducer of the right side in Figure 3. The main difference is that the tails are merged.

![Fig. 3. The transducers treeFSTCL (left) and optFSTCL (right).](image)

The construction and use of a tree like $treeFSTCL$ in Figure 3 is itself nothing new. However, the manipulation of graphs, trees and transducers has become quite simple in the theoretical framework of finite-state transducers implemented with software toolkits like the MIT transducer toolkit as used in [4]. Trees like $treeFSTCL$ will be used in Subsection 5.3 while the fully optimized networks are evaluated in Subsection 5.2.

4. EXPERIMENTAL SETUP

For the experiments to follow, we used an internal, name recognition corpus with telephone speech. All recognitions are large-vocabulary, isolated word recognition without language model to simplify the comparisons.

The test corpus contains 1282 utterances recorded over a variety of telephone lines. The statistical significance of the word error-rates at a 95% level is approximately 1%.

For training, we used 34.2 hours of speech material which contain 40,000 utterances. The training results in an acoustic model with about 100,000 Gaussian densities with density-specific variance.

The recognizer decodes speech based on a precompiled finite-state transducer network using a standard MFCC frontend. The finite-state transducer network maps triples of mixture-ids, which model triphones, to words and is built from a lexicon with 130,000 baseforms. Note that we do not use language model or pronunciation weights, i.e., there are no weights in the FST networks.

During the decoding, we expand the sequences of mixture-ids on-the-fly to hidden Markov models and apply histogram pruning to limit the search space to a given maximum of active state hypotheses. The experiments were performed on a variety of Intel Pentium III systems running Linux.

5. RESULTS

In a preliminary experiment, we compared the baseline LVCSR decoder [1] with the new decoder based on finite-
state transducers. In this experiment, the transducer $\text{treeTCL}$ is a tree which maps triples of mixture-ids to words and which mimics the phonetic prefix tree in the baseline decoder. As shown in Figure 4, the error-rates for baseline and FST decoder are identical.

5.1. Comparison of tree and optimized FSTs

After the confirmation that the FST decoder works as expected, we investigate more interesting finite-state transducers. First, we compute a maximal compact FST called $\text{optTCL}$ and compare that with $\text{treeTCL}$. This experiment shows how much we can reduce the search-effort when we use $\text{optTCL}$ instead of a standard phonetic tree. The result is given in Figure 4.

![Figure 4](image-url) A comparison of baseline, tree and optimized FSTs based on the number of active states and arcs.

Figure 4 shows that the fully optimized transducer $\text{optTCL}$ uses fewer active states while decoding. However, the difference compared to $\text{treeTCL}$ is small, except when the beam with active hypotheses is wide. For example, we achieve 8.5% error-rate with decoding $\text{optTCL}$ at $\approx 23k$ active hypotheses, while the baseline decoder uses 40k hypotheses. The best error-rate of almost 8% is achieved with 36k active hypotheses with $\text{optTCL}$ and 53k hypotheses for the baseline decoder. An improvement of 30% - 40%, i.e., the effective search space is 30% - 40% smaller in the FST case. For reference, we note that the preliminary experiments show that the asymptotic error-rate with a full search without histogram pruning gives 7.80% using about 200k active hypotheses.

5.2. More compact transducer networks

Although the $\text{optTCL}$ network in the previous section is the most compact FST given a TCL network, we have not yet exploited all sharing options. In the TCL transducer, the input symbols of every arc are triples of mixture-ids like $[1, 2, 3]$. If we build $\text{STCL}$ with one mixture-id per arc as input symbol, and then do factoring, we ‘push’ the labels to the front and concentrate as many common prefixes of mixture-ids onto every arc. The effect of factoring on the static FST sizes is given in Table 1 while the benefits of decoding are presented in Figure 5.

<table>
<thead>
<tr>
<th></th>
<th>$\text{treeTCL}$</th>
<th>$\text{optTCL}$</th>
<th>$\text{optFTCL}$</th>
<th>$\text{optFSTCL}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>#states</td>
<td>1304639</td>
<td>221078</td>
<td>109217</td>
<td>128055</td>
</tr>
<tr>
<td>#arcs</td>
<td>1581667</td>
<td>419391</td>
<td>306024</td>
<td>293816</td>
</tr>
<tr>
<td>#input</td>
<td>6161</td>
<td>6161</td>
<td>45649</td>
<td>31462</td>
</tr>
</tbody>
</table>

![Figure 5](image-url) The $\text{treeTCL}$ and $\text{optTCL}$ transducers compared to decoding with the factored $\text{optFTCL}$ and $\text{optFSTCL}$.

Figure 5 shows that decoding with the factored $\text{optFSTCL}$ model uses substantially fewer active states to achieve the error-rates of $\text{treeTCL}$. Compared to the baseline or $\text{treeTCL}$, the active search space of $\text{optFSTCL}$ has only half the number of active states of the baseline decoder to achieve the same error-rate.

For example, we achieve 8.5% error-rate with decoding $\text{optFSTCL}$ at $\approx 17k$ active hypotheses, while the baseline decoder uses 40k hypotheses. Alternatively, at a fixed number of active states, e.g., 20k, the error-rate of the decoder with the $\text{optFSTCL}$ transducer is 8.34% compared to 9% for the baseline decoder, i.e., about 8% relative better than the baseline.

In this experiment, the best achieved error-rate is 7.80% by the $\text{optFSTCL}$ decoder at 35k active states in Figure 5. Note that this is the same error-rate as the full search from Subsection 5.1 where we needed 200k active states. Therefore, the combination of FST algorithms can design a compact search space where we use only $\approx 20\%$ of the original number of active states to achieve the best possible error-rate.
5.3. From phonetic to factored tree

Finally, we change the structure of the \textit{treeTCL} transducer to get rid of redundancies in the tree. If we can build a factored tree which shares common prefixes of mixture-ids then even a standard LVCSR decoder can profit.

In a first experiment, we take the \textit{treeTCL} transducer from Subsection 5.1. Its arcs are always labeled with three mixture-ids to reflect the hidden Markov model. We know that the \textit{optFSTCL} transducer exploits best the common sequences of mixture-ids of any length, resulting in best performance for small beamwidth. However, the \textit{optFSTCL} has 31642 different labels with a varying number of mixture-ids, while the original \textit{treeTCL} has 6161 different mixture-ids sequences of length three.

Table 2. Size of the original tree-like transducer \textit{treeTCL} and the special, factored tree \textit{treeFSTCL}. The #mixture-ids denotes the average number of mixture-ids per input label over all arcs.

<table>
<thead>
<tr>
<th></th>
<th>\textit{treeTCL}</th>
<th>\textit{treeFSTCL}</th>
</tr>
</thead>
<tbody>
<tr>
<td>#states</td>
<td>1304639</td>
<td>1962246</td>
</tr>
<tr>
<td>#arcs</td>
<td>1581667</td>
<td>2239274</td>
</tr>
<tr>
<td>#input</td>
<td>6161</td>
<td>31462</td>
</tr>
<tr>
<td>#mixture-ids</td>
<td>2.91</td>
<td>1.85</td>
</tr>
</tbody>
</table>

Using FST tools, we build \textit{treeFSTCL} and compare with \textit{treeTCL} in Figure 6. \textit{treeFSTCL} is built by enumerating the words in \textit{optFSTCL}, reconstructing the lexicon based on the input symbols of \textit{optFSTCL}, and building a new tree. For comparison, we give the static sizes of the trees in Table 2. Note that the factored tree is larger than the original tree, but the sharing of mixture-ids prefixes is beneficial for a search space reduction of the Viterbi search. Also, \textit{treeFSTCL} combines common prefixes at the price of input labels with fewer mixture-ids.

Fig. 6. A comparison of tree transducer \textit{treeTCL} and factored \textit{treeFSTCL} with \textit{optTCL} and \textit{optFSTCL}.

First, we learn from Figure 6 that \textit{treeFSTCL} outperforms \textit{treeTCL}. Second, we learn that \textit{treeFSTCL} performs equally good compared to \textit{optTCL}. This means that suffix sharing in \textit{optTCL} and sharing common prefixes across hidden Markov models by factoring contribute in equal proportions to the improved error-rate for this large-vocabulary, isolated word recognition task.

6. CONCLUSION

We compared a LVCSR decoder [1] with a decoder based on finite-state transducers and investigated in detail the contribution of techniques like factoring and optimization of the FST to the reduction of the search space for this isolated word recognition task.

First, we find that suffix sharing by minimization and factoring contribute in about equal proportions to the reduction of the search space of the investigated task. Compared to the baseline decoder, we achieve the same error-rate with \textit{optFSTCL} with a much smaller beam of about 50% fewer active hypotheses. In addition, we achieve the best error-rate of the baseline decoder with 35k active state hypotheses using \textit{optFSTCL} instead of 200k active state hypotheses with the baseline, i.e., a reduction of a factor four. The interaction with other LVCSR issues, like crossword transitions, is currently investigated.

Finally, we showed that factoring a phonetic tree is a way to achieve about half of the search space reduction of the fully optimized transducer network, with the advantage that the factored tree can be used in a traditional LVCSR framework.

7. REFERENCES


I would like to thank my colleagues Xavier Aubert and Christoph Neukirchen, and Lee Hetherington from the SLS group at MIT, for many interesting discussions.