INCREMENTAL LANGUAGE MODELS FOR SPEECH RECOGNITION USING FINITE-STATE TRANSDUCERS

Hans J.G.A. Dolfing

Philips Research Laboratories
Weißhausstrasse 2
D-52066 Aachen, Germany
hans.dolfing@philips.com

I. Lee Hetherington

Spoken Language Systems Group
MIT Laboratory for Computer Science
Cambridge, MA 02139 USA
ilh@mit.edu

ABSTRACT

In the context of the weighted finite-state transducer approach to speech recognition, we investigate a novel decoding strategy to deal with very large n-gram language models often used in large-vocabulary systems. In particular, we present an alternative to full, static expansion and optimization of the finite-state transducer network. This alternative is useful when the individual knowledge sources, modeled as transducers, are too large to be composed and optimized. While the recognition decoder perceives a single, weighted finite-state transducer, we apply a divide-and-conquer technique to split the language model into two parts which add up exactly to the original language model. We investigate the merits of these 'incremental language models' and present some initial results.

1. INTRODUCTION

In the context of the weighted finite-state transducer approach to speech recognition, we investigate a novel decoding strategy. Recent papers [1–3] show that finite-state transducers are an attractive alternative for speech decoding. However, there is some concern with respect to the size of the transducers, especially in the light of large-vocabulary speech recognition tasks which involve very large language models. If the sum of parameters from acoustic model, hidden Markov model (HMM), lexicon, triphones, and language models, gets too large, we cannot optimize the composed finite-state transducer anymore, losing the principal advantage of concentrating the speech decoding into one compact, minimally sized transducer network.

A straight-forward solution is to employ multiple recognition passes, composing only a part of the available knowledge sources in the statically optimized transducer and applying the remainder in a second rescoring pass [2].

Because one of the main problems in speech decoding is to minimize search errors, multi-pass recognizers are potentially more vulnerable since they require tuning in every pass. A one-pass recognizer can employ all available knowledge sources at the same time which results in the best possible recognition result, at the price of a more complex decoding process. Therefore, many studies have explored one-pass decoders [1, 4].

Finite-state transducers have the potential of combining the early use of all available knowledge sources, within a well understood theoretical framework. However, the resulting networks must have a limited size to be optimized.

Inspired by [4–6] and based on the divide-and-conquer idea, we split the main decoding finite-state transducer into two parts. The first transducer is statically optimized and contains the acoustic model information, lexical knowledge, and the 'important' part of the language model. The second transducer part contains the remainder of the language model. The two parts are composed on-the-fly during decoding, resulting in the decoder operating on a single logical transducer representing all the knowledge sources, including the full language model. Because the language model is now distributed over the two component finite-state transducers, and the scores of the language model add up exactly to the scores in the original transducer, we refer to this approach as an 'incremental language model.' 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 beam width in decoding and best possible decoding results.

In Section 2, we briefly discuss the general properties of a finite-state transducer–based recognition system and then in Section 3 we describe the incremental language models. We investigate the novel decoding strategy based on the Jupiter system [7, 8] for weather information as explained in Section 4 and we report results in Section 5. Finally, we summarize our 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 transducer, and these transducers are composed together to form the single transducer to be used for recognition [3]. Within

This research was supported by DARPA under contract N66001-99-1-8904 monitored through Naval Command, Control, and Ocean Surveillance Center and under an industrial consortium supporting the MIT Oxygen Alliance.

the MIT SUMMIT system used for the experiments to follow, we typically use the weighted finite-state transducer $CPLG$ = $opt(C \circ P \circ L \circ G)$ for recognition, where G is an n-gram language model, L the lexicon, P is a set of phonological rules [9], and C adds context-dependent phonetic models (usually diphones). Here, the optimization operator $opt(\cdot)$ performs ϵ-removal, weight pushing, determinization,[1] and minimization.

3. INCREMENTAL LANGUAGE MODELS

Particularly with large vocabularies and large language models, computing $CPLG$ as in Section 2 can yield a transducer that is too large to be useful, even *if* it can be optimized. It has been observed that the size of this transducer is roughly proportional to the size of the language model [3]. Some large-vocabulary recognition tasks, e.g., broadcast news Hub-4 with 64,000-word vocabulary, can have trigram language models with in excess of 65 million parameters [10]. Thus, if we wish to build a one-pass recognition system using such a large language model, it is clear that we cannot precompose and optimize our $CPLG$. Even if we could construct it, it would be too large to fit into a reasonable amount of memory.

A straightforward solution would be to perform recognition on $opt(C \circ P \circ L) \circ G$, where the composition with G would be on-the-fly during decoding and $opt(C \circ P \circ L)$ is the statically optimized component. However, this would lose the advantages of having G inside the optimization. In particular, when G is composed inside the determinization, the language model scores become distributed or 'smeared' along the weights of the resulting transducer. As we will see in Section 5, this language model score smearing is advantageous within a decoder using beam-pruning [4, 5] and can be thought of as language model lookahead. At the beginning of a word, we may get a piece of the word's eventual language model score (along with all the scores of other words sharing the transitions). This tends to incorporate part of the language model sooner in the decoding process, and it also tends to smooth out the finite-state transducer weights. Without such language model smearing, a word's full language model contribution would be located on a single transition, and this sudden contribution could knock the word out of the decoder's search beam.

The question is how to get the advantages of language model smearing, or lookahead, without having to use the full, and possibly too large, language model within the statically optimized finite-state transducer. If we factor our language model as $G = G_s \circ G_i$, consisting of a smearing language model G_s and an *incremental* language model G_i, we can then perform recognition on

$$CPLG = opt(C \circ P \circ L \circ G_s) \circ G_i \ .$$

[1] In the experiments to follow, we used partial determinization of degree 4, meaning that we allow up to four transitions with the same input label to remain undeterminized. We find that this partial determinization often yields smaller transducers and better time/space tradeoffs.

G_s is a hopefully smaller language model to be used during the static optimization, and G_i is the desired full language model with its scores adjusted for the already applied G_s. Thus the relatively large G_i will be applied on-the-fly during decoding, correcting the net language model scores to be those of the desired G.

We can construct G_i from G by adjusting G's scores for those already applied by G_s. In particular, if the weights are log probabilities, for each G weight we subtract the appropriate G_s weight to form G_i. In our construction, G_i will have exactly the same topology as G; only its weights will be different. For comparison, the traditional approach to smearing in a LVCSR system [6] is to compute on demand the n-gram score distributions over the phonetic prefix tree. The proposed incremental language model which uses a general G_s for smearing is much more flexible and integrates well with a general finite-state transducer–based decoder.

In the case of backoff n-gram language models, care must be used when applying G_i. Generally, when we apply G_s, we allow backoff ϵ paths to compete with non-backoff paths. In constructing our n-grams, we can guarantee that all non-backoff paths always score better than their corresponding backoff paths. A problem arises when constructing G_i due to the *subtraction* of G_s scores: a backoff path within G_s that scores worse than a non-backoff path can result in a backoff path in G_i that scores better than its non-backoff counterpart. The subtraction of scores negates the sense of better scores. We avoid this problem by using G_i in a strictly deterministic manner. We have an n-gram-specific finite-state transducer type that we can put into deterministic mode to follow backoff only where strictly necessary. Thus, with G_i in deterministic mode we can apply it and guarantee that the net language model is the desired G.

In summary, by factoring our language model $G = G_s \circ G_i$, we have the flexibility to explore how much complexity to put into the statically optimized component $opt(C \circ P \circ L \circ G_s)$ and how much to leave to the dynamically applied incremental language model component G_i.

4. EXPERIMENTAL SETUP

For the experiments to follow, we used the recognition task of the Jupiter conversational weather information system [8]. The test corpus contains 1711 utterances, which accumulates to 9659 words and 1.6 hours of data. The data is recorded over a variety of telephone lines. The statistical significance of the reported word error-rates at a 95% level is approximately 0.6%.

For training, we used 116,867 utterances totalling 105 hours. The training results in an acoustic model with almost 30,000 Gaussian densities with density-specific variance. The preprocessing employs segmental modeling. The lexicon contains approximately 2000 words which we model with diphones.

The recognizer decodes speech based on a precompiled finite-state transducer network. The finite-state transducer network maps diphones to word sequences and was built

$\lvert G_s \rvert$	n-gram				
	1	2	3	4	5
$\emptyset$	27.4	13.9	13.2	12.6	12.7
1	–	11.0	10.2	9.8	9.8
2	–	–	9.7	9.2	9.3
3	–	–	–	9.2	9.2
n	27.3	10.6	9.7	9.2	*

(a) Beam width = 1000 nodes.

$\lvert G_s \rvert$	n-gram				
	1	2	3	4	5
$\emptyset$	27.4	11.3	10.6	10.1	10.0
1	–	10.6	9.5	9.0	9.0
2	–	–	9.3	8.8	8.8
3	–	–	–	8.8	8.7
n	27.3	10.5	9.3	8.8	*

(b) Beam width = 5000 nodes.

Table 1. Comparison of word error-rates [%] for various G and G_s. $|G_s|$ is the order m of the m-gram smearing language model G_s. $|G_s| = \emptyset$ corresponds to the full language model applied on-the-fly (i.e., $G_i = G$), and $|G_s| = n$ corresponds to the full language model statically composed and optimized into $CPLG$ (i.e., $G_s = G$). All language models are unpruned here. Tables (a) and (b) differ in the beam width (number of active nodes) used for decoding. * indicates we could not build the static $CPLG$ with the 5-gram.

from a lexicon, a set of phonological rules, and a language model as described in [7]. During the decoding, we apply histogram pruning on the number of active nodes, e.g., we limit the search space to 5000 active nodes. The experiments were performed on a variety of Intel Pentium III systems running Linux, including one with 2GB RAM for optimizing the largest transducers.

5. RESULTS

Because we want to compare the behavior of $CPLG$ with that of $CPLG_s \circ G_i$, we investigate the effects when G_s is empty, or when G_s is a full n-gram model with $n \geq 1$. In a second experiment, we prune the n-gram language models with various count thresholds, and use these pruned language models as G_s. In all experiments, the invariant is $G = G_s \circ G_i$. Note that the full, statically expanded network is $CPLG$ while the on-the-fly composition with the complete language model G is $CPL \circ G$ with an empty G_s.

In a first experiment, we compare incremental language models where G_s is a complete n-gram model. Table 1 shows us that decoding with the full, statically optimized network generally achieves the best error-rate. However, if G_s is a bigram or higher-order language model then we achieve the same error-rate as with the statically optimized network. If G_s is a unigram model then the decoding achieves a better error-rate compared to an empty G_s. However, it does not achieve full performance.

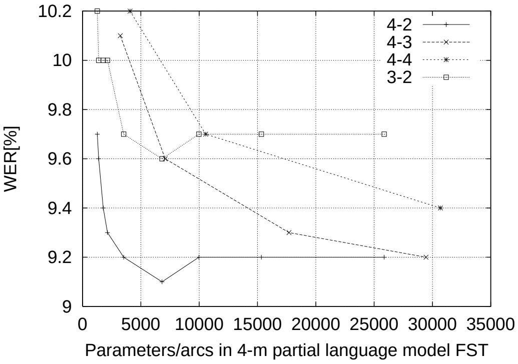

Fig. 1. Comparison of word error-rates in [%] with the number of parameters (arcs) of a pruned G_s. The notation *n-m* means an *n*-gram G and an *m*-gram G_s. Actually, the 4-*m* curves start at an error-rate of 12.6% and converges to 9.2%. For 3-*m*, the curve goes from 13.2% to a best error-rate of 9.7%. Pruning threshold 1000 nodes.

$\lvert G_s \rvert$	G_s		$opt(CPLG_s)$	
	states	arcs	states	arcs
$\emptyset$	0	0	7,082	23,026
1	1	1,232	7,082	23,026
$2'$	793	6,816	26,771	110,434
2	1,225	25,863	49,968	336,860
3	24,634	119,185	436,535	1,912,475
4	94,549	304,208	1,948,391	12,964,061
5	209,659	561,288	*	*

Table 2. Sizes of various n-gram language models and resulting $CPLG$s. $|G_s| = 2'$ indicates a bigram pruned to keep only those bigrams that occurred at least 8 times.

Second, we test the effect when G_s is a pruned language model instead of the full language model as in Table 1. Figure 1 shows that at an equal number of parameters for the G_s model, the 4-2 decoding (static bigram and incremental 4-gram) is most effective.

Interestingly, both 4-2 and 3-2 decoding exhibit a minimum error-rate for a bigram G_s of around 7000 arcs. In Table 2, this is the $|G_s| = 2'$, which is approximately 21% of the full bigram. The error-rate difference of 0.1% with respect to the error-rate of the full, statically optimized transducer, is not significant. Future experiments will have to study this in more detail, as well as investigate information-theoretic methods to generate the smearing language model.

Finally, we investigate the space/time properties of decoding with incremental language models. A subset of 50 utterances was used in this test. The results are summarized in Table 3 and Table 4, which show that there is a classic trade-off. The decoding is faster when more memory is used, i.e., to represent the fully optimized transducer. However, bigram pruned ($2'$), containing all bigrams that occurred at least 8

$\lvert G_s \rvert$	n-gram		
	2	3	4
$\emptyset$	1.03	1.04	1.04
1	0.89	0.93	0.98
$2'$	–	0.88	0.86
2	–	0.94	0.95
3	–	–	0.95
n	0.61	0.65	0.76

(a) Beam width = 1000.

$\lvert G_s \rvert$	n-gram		
	2	3	4
$\emptyset$	2.62	2.93	2.93
1	1.65	1.87	1.89
$2'$	–	1.71	1.66
2	–	1.86	1.81
3	–	–	1.92
n	0.87	1.05	1.23

(b) Beam width = 5000.

Table 3. Decoding speed (xRT) on 1.5GHz Pentium 4.

$\lvert G_s \rvert$	n-gram		
	2	3	4
$\emptyset$	58	60	63
1	57	59	62
$2'$	–	63	66
2	–	71	74
3	–	–	132
n	67	125	457

(a) Beam width = 1000.

$\lvert G_s \rvert$	n-gram		
	2	3	4
$\emptyset$	69	71	75
1	64	68	72
$2'$	–	75	78
2	–	84	87
3	–	–	150
n	76	141	478

(b) Beam width = 5000.

Table 4. Peak memory use (MB).

times, represents a particularly attractive operating point.

If we use the incremental language model approach, the memory savings are substantial. Compared to the fully optimized, static 4-gram finite-state transducer, the 4-$2'$ decoder uses less than 20% of the memory, achieves the same accuracy, but runs only 35% slower. Additionally, Table 2 shows that the static transducer is more than 100 times smaller compared to the fully optimized, static 4-gram transducer.

Also note that decoding speed is not inversely proportional to the number of arcs in the fully optimized finite-state transducer. For example, the decoding with the static 4-gram transducer is only 25–40% slower compared to the static bigram transducer, despite the fact that the 4-gram network is 10 times larger.

In addition, we conducted an experiment where the static network is fully determinized instead of the partial determinization (degree 4). This has only minor effects on the decoding speed but results in significantly larger transducers. Thus our use of partial determinization when statically optimizing.

6. CONCLUSION

We have introduced the use of incremental language models as a way to combat the very large weighted finite-state transducers that can result with large vocabularies and large language models. We factor the desired language model into two components, a relatively small component to be statically combined and optimized with other system components, and a larger incremental language model to be utilized during decoding. The flexibility provided by this factoring allows us to choose different operating points depending on memory or speed needs. By using on-the-fly composition, we combine the two components in a one-pass decoder.

By varying the complexity of the smearing or lookahead language model, we discovered that we can achieve optimal recognition accuracy with a strategy of statically composing and optimizing with a relatively small pruned bigram and dynamically composing with a 4-gram incremental language model on-the-fly. This strategy allows for optimal accuracy within a decoder with beam pruning while greatly reducing memory use as compared with using a single transducer constructed directly with the 4-gram. Speed is reduced modestly, at least when the desired language model is relatively large.

Finally, although we introduced the technique of incremental language models primarily to deal with large vocabularies and very large language models, we have performed the initial experiments with the medium-vocabulary Jupiter system that we are very familiar with. We have performed some preliminary experiments with a 25,000-word large vocabulary, continuous speech recognition task, and the results very closely mirror the findings presented in this paper. We intend to further investigate this technique with large vocabularies in the future.

7. REFERENCES

[1] X.L. Aubert, "A brief overview of decoding techniques for large vocabulary continuous speech recognition," in *Proc. Automatic Speech Recognition workshop 2000*, Paris, France, Sept. 2000, vol. 1, pp. 91–96.

[2] A. Ljolje, M. Riley, D. Hindle, and R. Sproat, "The AT&T LVCSR-2000 System," in *Proc. of the NIST Large Vocabulary Conversational Speech Recognition Workshop*, Maryland, May 2000.

[3] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," in *Proc. Automatic Speech Recognition workshop 2000*, Paris, France, Sept. 2000, vol. 1, pp. 97–106.

[4] X.L. Aubert, "One pass cross word decoding for large vocabularies based on a lexical tree search organization," in *Proc. European Conference on Speech Communication and Technology*, Budapest, Hungary, Sept. 1999, vol. 4, pp. 1559–1562.

[5] X.L. Aubert, C. Dugast, H. Ney, and V. Steinbiss, "Large vocabulary continuous speech recognition of Wall Street Journal data," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Adelaide, Australia, Apr. 1994, vol. 2, pp. 129–132.

[6] S. Ortmanns, H. Ney, and A. Eiden, "Language model lookahead for large vocabulary speech recognition," in *Proc. International Conference on Spoken Language Processing*, Philadelphia, PA, Oct. 1996, pp. 2095–2098.

[7] J. Glass, T.J. Hazen, and I.L. Hetherington, "Real-time telephone-based speech recognition in the Jupiter domain," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Phoenix, AZ, Mar. 1999, vol. 1, pp. 61–64.

[8] V. Zue et al., "Jupiter: A telephone-based conversational interface for weather information," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 85–96, Jan. 2000.

[9] I. L. Hetherington, "An efficient implementation of phonological rules using finite-state transducers," in *Proc. European Conference on Speech Communication and Technology*, Aalborg, Denmark, Sept. 2001.

[10] D. Klakow, X. Aubert, P. Beyerlein, R. Haeb-Umbach, M. Ullrich, A. Wendemuth, and P. Wilcox, "Language model investigations related to broadcast news," in *Proc. DARPA Broadcast News and Transcription Workshop*, Lansdowne, VA, Feb. 1998.