Pre-Initialized Composition For Large-Vocabulary Speech Recognition

Cyril Allauzen, Michael Riley

Google Research, 76 Ninth Avenue, New York, NY, USA

allauzen@google.com, riley@google.com

Abstract

This paper describes a modified composition algorithm that is used for combining two finite-state transducers, representing the context-dependent lexicon and the language model respectively, in large vocabulary speech recogntion. This algorithm is a hybrid between the static and dynamic expansion of the resultant transducer, which maps from context-dependent phones to words and is searched during decoding. The approach is to pre-compute part of the recognition transducer and leave the balance to be expanded during decoding. This method allows for a fine-grained trade-off between space and time in recognition. For example, the time overhead of purely dynamic expansion can be reduced by over six-fold with only a 20% increase in memory in a collection of large-vocabulary recognition tasks available on the Google Android platform.

Index Terms: WFST, LVCSR

1. Introduction

Weighted finite-state transducers (WFST)s are a commonly used representation in speech recognition [1]. They can represent a language model G (an automaton over words), the phonetic lexicon (L) (a context-independent (CI) phone-to-word transducer), and the context-dependency (CD) specification C (a CD-phone to CI-phone transducer). These models can be optimized by determinization and combined by finite-state composition. For example,

$$CL = C \circ Det(L) \tag{1}$$

$$T = CL \circ G \tag{2}$$

builds an efficient recognition transducer, mapping from context-dependent phones to words, that can be searched during ASR decoding [2].

This construction has been used principally in two ways. The recognition transducer can be fully constructed ahead of time *statically*. This is very time-efficient in use, but requires the most space. Alternatively, the composition in Equation 2 can be constructed *dynamically* (or *on-the-fly* or *lazily*). In this way, only the relatively small part of the composition that is visited during the recognition of an utterance needs to be created. This can save considerable space since often $|CL| + |G| \ll |CL \circ G|$ but uses more time during recognition. The dynamic approach also permits G to be modified prior to recognition of an utterance without requiring the time-consuming full reconstruction of $CL \circ G$. For example, a grammar non-terminal for personal contacts might be replaced with the current user's actual contacts just before recognition. In this paper, we describe an approach that is a hybrid between static and dynamic composition, which allows a finer trade-off between time and space. Caseiro and Trancoso [3] used a specific hybrid composition based on topological features on the recognition transducer in earlier work.

2. Composition Algorithm

A detailed description of weighted finite-state transducers - their theory, algorithms and applications to speech recognition - is given in [1]. We present only those concepts here that are needed to describe our algorithms.

2.1. Preliminaries

A semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is ring that may lack negation. If $\otimes$ is commutative, we say that the semiring is *commutative*.

The *probability semiring* $(\mathbb{R}_+, +, \times, 0, 1)$ is used when the weights represent probabilities. The *log semiring* $(\mathbb{R} \cup \{-\infty, +\infty\}, \oplus_{\log}, +, \infty, 0)$, isomorphic to the probability semiring via the negative-log mapping, is often used in practice for numerical stability. The *tropical semiring* $(\mathbb{R} \cup \{-\infty, +\infty\}, \min, +, \infty, 0)$, derived from the log semiring using the *Viterbi approximation*, is often used in shortest-path applications.

A *weighted finite-state transducer* $T = (\mathcal{A}, \mathcal{B}, Q, I, F, E, \lambda, \rho)$ over a semiring $\mathbb{K}$ is specified by a finite input alphabet $\mathcal{A}$, a finite output alphabet $\mathcal{B}$, a finite set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, a finite set of transitions $E \subseteq Q \times (\mathcal{A} \cup \{\epsilon\}) \times (\mathcal{B} \cup \{\epsilon\}) \times \mathbb{K} \times Q$, an initial state weight assignment $\lambda : I \to \mathbb{K}$, and a final state weight assignment $\rho : F \to \mathbb{K}$. $E[q]$ denotes the set of transitions leaving state $q \in Q$.

Given a transition $e \in E$, $p[e]$ denotes its origin or previous state, $n[e]$ its destination or next state, $i[e]$ its input label, $o[e]$ its output label, and $w[e]$ its weight. A *path* $\pi = e_1 \cdots e_k$ is a sequence of consecutive transitions: $n[e_{i-1}] = p[e_i]$, $i = 2, \ldots, k$. The functions n, p, and w on transitions can be extended to paths by setting: $n[\pi] = n[e_k]$ and $p[\pi] = p[e_1]$ and by defining the weight of a path as the $\otimes$-product of the weights of its constituent transitions: $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$. A *string* is a sequence of labels; ϵ denotes the empty string.

The weight associated by T to any pair of input-output strings (x, y) is defined as:

$$T(x, y) = \bigoplus_{\pi \in \cup_{q \in I,\, q' \in F} P(q, x, y, q')} \lambda[p[\pi]] \otimes w[\pi] \otimes \rho[n[\pi]], \tag{3}$$

where $P(q, x, y, q')$ denotes the set of paths from q to q' with input label $x \in \mathcal{A}^*$ and output label $y \in \mathcal{B}^*$.

Let $T_{|R} = (\mathcal{A}, \mathcal{B}, Q_R, I_R, F_R, E_R, \lambda_R, \rho_R)$ denote a restriction of the transducer T to $R \subseteq Q$ with $Q_R = R \cup \{n[e] \,|\, e \in E \wedge p[e] \in R\}$, $I_R = I \cap R$, $F_R = F \cap R$, $E_R = \{e \in E \,|\, p[e] \in R\}$, $\lambda_R = \lambda : I_R \to \mathbb{K}$ and $\rho_R = \rho : F_R \to \mathbb{K}$. Thus R restricts the states from which transitions may exit.

2.2. Composition

Let $\mathbb{K}$ be a commutative semiring and let T_1 and T_2 be two weighted transducers defined over $\mathbb{K}$ such that the input alphabet $\mathcal{B}$ of T_2 coincides with the output alphabet of T_1. The result of the composition of T_1 and T_2 is a weighted transducer denoted by $T_1 \circ T_2$ and specified for all x, y by:

$$(T_1 \circ T_2)(x, y) = \bigoplus_{z \in \mathcal{B}^*} T_1(x, z) \otimes T_2(z, y). \tag{4}$$

Leaving aside transitions with ϵ inputs or outputs, the following rule specifies how to compute a transition of $T_1 \circ T_2$ from appropriate transitions of T_1 and T_2: (q_1, a, b, w_1, q_1') and (q_2, b, c, w_2, q_2') results in $((q_1, q_2), a, c, w_1 \otimes w_2, (q_1', q_2'))$. A simple algorithm to compute the composition of two ϵ-free transducers, following the above rule, is given in [1].

2.3. Pre-Initialized Composition

Suppose we wish to compute the composition $T = T_1 \circ T_2$ but already have $T_{|R}$, a portion of that composition, in hand. Under these circumstances, we can modify the simple composition algorithm so that it is pre-initialized by $T_{|R}$, saving its construction. Figure 1 gives this modified algorithm. It differs from standard composition by initializing the set of states, the initial state, the final states and transitions from those in $T_{|R}$ at lines 1-4 and by skipping computing the transitions leaving a state already in $T_{|R}$ at line 9 instead finding any successor states that need to be enqueued directly from $T_{|R}$ at lines 20-22.

In the simplest implementation Q, F, and E are represented as sets with INSERT(Q, q) defined as $Q \leftarrow Q \cup \{q\}$ and so forth. However, we wish the initialization step in lines 1-3 to be constant time in our implementation and we would like to share $T_{|R}$ in parallel calls to the composition algorithm in multiple threads. Therefore we will represent $Q = (Q_{static}, Q_{dyn})$ as a pair of sets denoting a *static* and *dynamic* part. Then the initialization on line 1, $Q \leftarrow Q_R$ becomes $Q \leftarrow (Q_R, \phi)$, INSERT(Q, q) is defined as $Q \leftarrow (Q_{static}, Q_{dyn} \cup \{q\})$ and $q \in Q$ means $q \in Q_{static} \vee q \in Q_{dyn}$. We use similar data structures for F and E. In this way, the static part is set only at initialization (and can be shared among parallel calls) while the dynamic part is modified during the running of algorithm's iterations.

While we have presented pre-initialized composition for the simple case with no *composition filter*, the algorithm extension to allow for different filters that handle epsilon transitions and various forms of lookahead can be similarly modified [4, 2].

This algorithm was implemented in `OpenFst` [5], a C++ library for weighted finite-state transducers. In that library the composition algorithm is templated on the data structures that represent Q, F, E and the composition filter. In the standard case, simple set and filter representations are the default. For the speech recognition version that implements the composition in Equation 2, more complex set represention, with distinct static and dynamic components as outlined above are used and a lookahead filter ensuring efficient matching is also used [2].

In the speech recognition setting, $T_{|R}$ represents the prebuilt, static part of the recognition transducer that is shared among all utterances that are decoded. The balance of the recognition transducer needed for each utterance is built dynamically and discarded at the end of each utterance. Since decoding employs pruning (because of the very large search space), only a portion of T is visited during each utterance. Which states are visited in decoding and the order in which they are visited specifies the queue *discipline* for a given input in the algorithm in Figure 1 and if we allow the queue to discard states, the pruning can be modeled as well.

WEIGHTED-COMPOSITION$(T_1, T_2, T_{|R})$
1 $Q \leftarrow Q_R$
2 $F \leftarrow F_R$
3 $E \leftarrow E_R$
4 $I \leftarrow I_R$
5 $S \leftarrow I$
6 **while** $S \neq \emptyset$ **do**
7 $(q_1, q_2) \leftarrow$ HEAD(S)
8 DEQUEUE(S)
9 **if** $(q_1, q_2) \notin R$ **then**
10 **if** $(q_1, q_2) \in F_1 \times F_2$ **then**
11 INSERT$(F, (q_1, q_2))$
12 $\rho(q_1, q_2) \leftarrow \rho_1(q_1) \otimes \rho_2(q_2)$
13 **for** each $(e_1, e_2) \in E[q_1] \times E[q_2]$ s. t. $o[e_1] = i[e_2]$ **do**
14 **if** $(n[e_1], n[e_2]) \notin Q$ **then**
15 INSERT$(Q, (n[e_1], n[e_2]))$
16 ENQUEUE$(S, (n[e_1], n[e_2]))$
17 $w' \leftarrow w[e_1] \otimes w[e_2]$
18 INSERT$(E, ((q_1, q_2), i[e_1], o[e_2], w', (n[e_1], n[e_2])))$
19 **else**
20 **for** each $e \in E_R[(q_1, q_2)]$ **do**
21 **if** $n[e] \notin R$ **then**
22 ENQUEUE$(S, n[e])$
23 **return** T

Figure 1: Pseudocode of the composition algorithm with pre-initialization.

3. Initialization Methods

Modifying composition to permit pre-initialization by a transducer $T_{|R}$, as described above, is quite straight-forward. What remains to decide is the *initialization set* $R \subseteq Q$. Two trivial candidates are $R = I$, which uses fully-dynamic expansion, and $R = Q$, which uses a fully-static $T = T_R$. We now describe two other ways to select the initialization set R.

3.1. State Statistics

One natural candidate for the set R uses the state frequencies observed during the decoding of utterances representative of those expected in the future. Given an utterance μ_i, let $O_i \subseteq Q$ denote those states that are enqueued during the recognition of μ_i in the fully-dynamic version ($R = I$) of the algorithm in Figure 1. Given M utterances and a count threshold n, we can choose the initialization set as:

$$N(q) = \sum_{i=1}^{M} \mathbf{1}_{q \in O_i} \tag{5}$$

$$R = \{q \in Q \mid N(q) \geq n\} \tag{6}$$

$N(q)$ counts the number of utterances in which state q was enqueued and R contains all states whose count meets the threshold. Various values of n can be used to trade-off time versus space.

3.2. State Probabilities

Another natural approach for the set R uses the state probabilities intrinsic to the lexical and language models. The language model G is typically a stochastic n-gram model trained on text; the lexical model, if it has alternative pronunciations, may also be probabilistic.

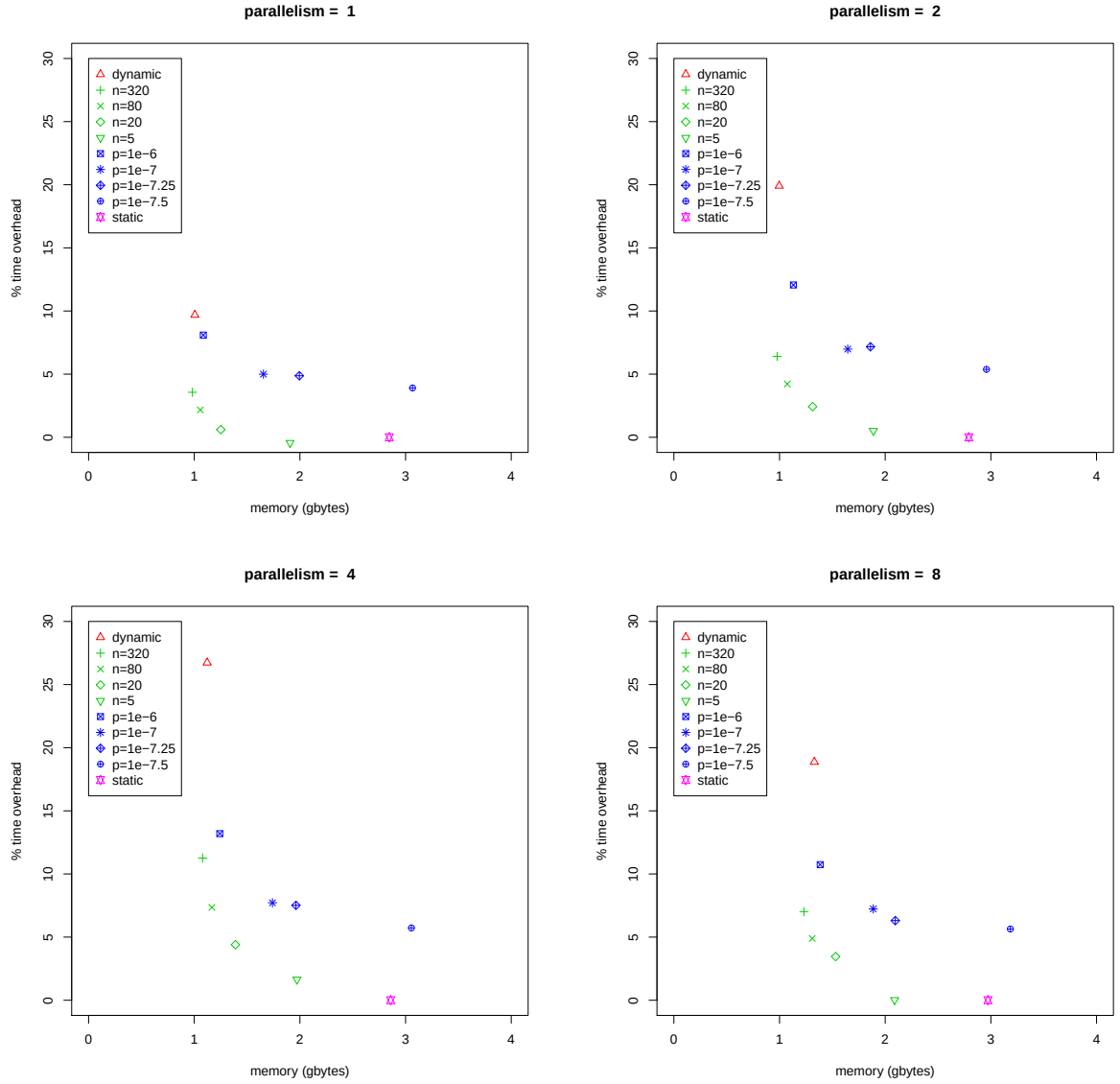

Figure 2: Total CPU memory usage (RAM) versus percent total recognition time in excess of that with fully-static composition with the same 1000 utterances served from one, two, four and eight threads for various recognition transducer expansion methods. The magenta six-sided star shows fully static case, the red triangle shows the fully-dynamic case, the blue points show the statistics-based method for various count cutoffs and the green points show the the probability-based method for various probability cutoffs.

First, assume that T is *stochastic:* $\mathbb{K}$ is the probability (or log) semiring and $\forall q \in Q$, $\bigoplus_{e \in E[q]} w[e] \oplus \rho(q) = \overline{1}$. In other words, the sum of all weights leaving a state sums to $\overline{1}$. In that case, the *shortest distance* from I to q, defined as:

$$\delta(q) = \bigoplus_{\pi \in P(q)} w[\pi] \tag{7}$$

where $P(q)$ is the set of paths from I to q, is the probability (or negative log probability) of reaching state q from I according to the stochastic distribution [6]. Given a threshold p, we can the choose the initialization set as:

$$R = \{q \in Q \mid \delta(q) \geq p\}. \tag{8}$$

Various values of p can be used to trade-off time versus space. In the case where T is not stochastic, it can be first *pushed* to make it so [1]. Both shortest distance and weight pushing are algorithms available in `OpenFst`.

4. Experiments

In this section, we report experimental results comparing the different recognition transducer expansion strategies.

4.1. Data, Models, and Methods

Our approach is evaluated on a randomized, anonymized utterance sampling of Voice tasks available on the Google Android platform. Frequent tasks include SMS, Voice Search, Google Maps, and Facebook. Recognition error rates are reported on a test set that consists of 27,327 transcribed utterances of 153,796

words. The speed and memory tests are reported on a random 1000 utterance subset. The state frequency statistics collection, described in Section 3.1, was performed on a separate 10,000 utterance subset.

The acoustic model used in our experiments is a DNN/HMM hybrid model whose input features are the concatenation of 26 consecutive frames of 40-dimensional log filterbank energies [7].

The language model is a 5-gram model obtained by interpolating, using the bayesian method in [8], a dozen individually-trained Katz-backoff n-gram language models from distinct data sources [9]. The sources include typed data sources (such as web search queries or SMS) and spoken data sources consisting of ASR results from anonymized utterances which have been filtered by their recognition confidence score. The data sources used vary in size, from a few million to a few billion sentences, making a total of 7 billion sentences. The language models are pruned using the relative entropy method of [10]. The resulting baseline language model G has 22.5 million n-grams (1.2 million unigrams, 10.1 million bigrams, 8.1 million 3-grams, 2.6 million 4-grams and 0.5 million 5-grams) represented as an FST with 3.2 million states and 30 million transitions. The context-dependent lexicon CL has 1.4 million states and 3.4 million transitions; the fully-expanded recognition transducer T has 45 million states and 98 million transitions.

Decoding was performed using a multi-threaded server that allows processing of multiple utterances simultaneously and a multi-threaded load tester that supplied the test utterances at a specified parallelism.[1] Dynamic expansion involves work that cannot be shared, impacting both time and space with increasing parallelism.[2] Results are reported with a range of parallelism since it is important factor to evaluate in practical systems that must serve users at scale. The pruning beam was set so that there are few search errors, comparable to our production settings. All timings were performed on an HP Z600 workstation with 24 cores and 50 gbytes of RAM.

4.2. Results

Figure 2 shows the memory versus recognition time trade-offs with the various methods of recognition transducer expansion described above using the baseline language model. If t_m is the time taken to recognize the 1000 test utterances with method m and t_{static} is the time taken to recognize with the pre-built $CL \circ G$, the vertical axis is $100 * (t_m - t_{static})/t_{static}$, the percent total recognition time in excess of that with the fully-static composition.

This figure shows that the fully-dynamic approach uses about one-third the memory of the fully-static approach but has about 10% overhead relative to that with no parallelism and about 27% overhead with a parallelism of four.

The composition pre-initialization using states frequently seen in recognition on a held-out 10,000 utterance set, offers a range of time-memory operating points depending on the utterance count cutoff n. For example with $n = 20$ and a parallelism of four, the excess recognition time is cut by over a factor of six compared to fully dynamic expansion with only about a 20% increase in RAM usage. With $n = 20$ about 1 million states are present in the initialization set R or about 2% of the fully-expanded T.

[1]The recognition server itself uses multiple threads per utterance with the acoustic model and search in separate threads.

[2]This implies with a small R and multiple threads, the hybrid approach might use less memory than the fully dynamic one. With large R, the hybrid approach might use more memory than the fully static case due to storing CL, G and most of $CL \circ G$.

n-grams	WER	% time overhead		$\|R\|$
		fully-dynamic	$n = 20$	$n = 20$
12,324,719	11.0%	26.4%	6.7%	1,017,531
22,329,335	10.7%	28.5%	4.3%	1,034,218
45,280,869	10.4%	31.8%	7.3%	1,027,841
70,594,261	10.4%	36.3%	6.5%	1,009,975

Table 1: Number of n-grams, word error rate and percent total recognition time in excess of that with fully-static composition with utterances served from four threads for various language model sizes with fully-dynamic expansion and with an utterance count cutoff of $n = 20$ for states in R. (Note the second LM is quite similar but not identical in size or results to the baseline model used for Figure 2 due to small differences in the experimental setups.)

The pre-initialization approach using the probabilities intrinsic in the language model also provides a range of time-memory operating points but in general performs worse than the approach based on recognition state statistics.

Table 1 shows the affect of the language model size both when the pre-initialization set is I and when it is chosen using the frequently seen states method with a utterance count cutoff of $n = 20$ using a parallelism of four in testing. The excess recognition time compared to the fully-static construction grows with increasing language model size for the the fully-dynamic case. Using the frequently seen states method with $n = 20$, the number of states in R is roughly constant and the time overhead varies between about 4-7% with no clear pattern with respect to LM size. The smaller LMs are derived from the largest through relative entropy pruning [10].

5. Discussion

Based on these results, the hybrid approach advocated here clearly offers operating points with large reductions in decoder overhead from fully-dynamic expansion with only modest increases in memory especially with increasing parallelism. In general, all the dynamic approaches show increasing overhead with increasing parallelism up to four in our experiments; this trend is expected due to thread contention in the transducer expansion. Expansion overhead decreases some with a parallelism of eight; this is presumably because other contentions begin to dominate.

It is not surprising that the state frequency initializaion method is better than the intrinsic state probability method in our experiments. The statistics were collected from utterances that were well matched to the test set while the language model is drawn from a wider range of sources. Further, the statistical method can take into account acoustic confusions unlike the probabilistic method.

With the state frequency initialization method, the result that the time overhead is a weak function of the language model size suggests the overhead for a given initialization set size is primarily determined by the underlying corpus on which the LM was trained.

The fully-dynamic method can obviously be used when the grammar G is modified just prior to recognition. The hybrid composition expansion methods can also be used so long as the initialization set R is restricted to states in the original grammar (i.e. (q_1, q_2) such that q_2 is in the unmodified portion of G).

6. References

[1] M. Mohri, F. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," in *Handbook of Speech Processing*, Y. H. Jacob Benesty, Mohan Sondhi, Ed. Springer, 2008, pp. 559–582.

[2] C. Allauzen, M. Riley, and J. Schalkwyk, "A generalized composition algorithm for weighted finite-state transducers," in *Proc. of Interspeech*, 2009, pp. 1203–1206.

[3] D. Caseiro and I. Trancoso, "Transducer composition for on-the-fly lexicon and language model integration," in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2001.

[4] T. Oonishi, P. Dixon, K. Iwano, and S. Furui, "Implementation and evaluation of fast on-the-fly WFST composition algorithms," in *Proc. Interspeech*, 2008, pp. 2110–2113.

[5] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *CIAA*, ser. LNCS, vol. 4783, 2007, pp. 11–23, `http://www.openfst.org`.

[6] M. Mohri, "Semiring Frameworks and Algorithms for Shortest-Distance Problems," *Journal of Automata, Languages and Combinatorics*, vol. 7, no. 3, pp. 321–350, 2002.

[7] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proc. Interspeech*, 2012.

[8] C. Allauzen and M. Riley, "Bayesian language model interpolation for mobile speech input," in *Proc. Interspeech*, 2011.

[9] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recogniser," *IEEE Transactions on Acoustic, Speech, and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987.

[10] A. Stolcke, "Entropy-based pruning of backoff language models," in *Proc. of DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 270–274.