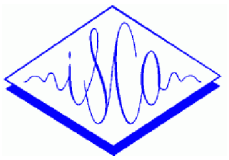
ISCA Archive
http://www.isca-speech.org/archive



INTERSPEECH 2012
ISCA's 13th Annual Conference
Portland, OR, USA
September 9-13, 2012

A Specialized WFST Approach for Class Models and Dynamic Vocabulary

Paul R. Dixon *Chiori Hori* *Hideki Kashioka*

National Institute of Information and Communications Technology, Japan

paul.dixon@nict.go.jp

Abstract

In this paper we describe a specialized Weighted Finite State Transducer (WFST) framework for handling class language models and dynamic vocabulary in automatic speech recognition. The proposed framework has several important features, a fused composition algorithm that substantially reduces the memory usage in comparison to generic WFST operations, and an efficient dynamic vocabulary scheme that allows for arbitrary new words to be added to class based language models on-the-fly without requiring any changes to the pre-compiled transducers. The dynamic vocabulary approach achieves very low run-time costs by representing the dynamic vocabulary items inserted into the language model from an optimum set of existing lexicon items. Experimental results on a voice search task illustrate the low runtime costs of the proposed approach.

Index Terms: speech recognition, decoding, WFST

1. Introduction

In recent years the Weighted Finite State Transducer (WFST) based approach to speech recognition search space construction and decoding [8] has become very popular. One of the advantages of the WFST framework is the formal manner each of the knowledge sources can be represented in a consistent fashion. These knowledge sources can then be composed together and optimized for speed and size ahead of decoding.

The recognition cascade is constructed from the following components; the Language Model G which represents the recognition grammar, the lexicon L which is built from the pronunciation dictionary and maps from phoneme sequences to words, a transducer C that converts context-dependent phonemes to context-independent phonemes, and optionally the acoustic models H. Our decoder performs the expansion of H dynamically and searches a cascade that is constructed according to:

$$\pi(C \circ det(L \circ G)) \quad (1)$$

where det is the determinization operation, $\circ$ is the composition operation and the π operator is a procedure that removes auxiliary symbols.

The static search network approach allows for the construction of very fast decoders. However, many modern speech recognition tasks require online changes to the knowledge sources such as adaptation of the models or extension of the vocabulary. In the WFST framework this becomes difficult because of the time and memory costs required to create the static networks. Recently composition algorithms have been developed [2] that allow very efficient online incorporation of the language model, however, these algorithm are most suitable when the vocabulary of L is fixed.

In this paper we address the problem of dynamically adding new vocabulary to WFST cascades. The proposed framework has several important features: a fused composition algorithm that reduces the memory usage in comparison to generic WFST operations and an efficient dynamic vocabulary scheme that allows for arbitrary new words to be added to class based language models on-the-fly without requiring any changes to the pre-compiled transducers. To achieve efficient decoding we describe a WFST based algorithm that selects an optimal set of in-vocabulary items that match the pronunciation of the dynamic vocabulary. By performing a unit selection that favours longer matches from the pre-compiled lexicon we show that there is negligible performance and memory overheads in comparison to a WFST cascade containing full vocabulary coverage.

1.1. Previous Work

A dynamic grammar approach was proposed in [13] that allowed for sub-grammars to be spliced into a main recognition grammar. The algorithm included a specialized scheme to correctly handle the cross word context dependency of the spliced grammars. This approach was extended by Hetherington [7] to a multi-pass scheme that also allowed for dynamic vocabulary. The similarity with our algorithm is the dynamic vocabulary items are added to n-gram classes online.

2. Implementation

In our task we are given a set of precompiled transducers and set of new probabilistic class to words mappings. In the case a new word does not exist in the base lexicon L we assume a set of pronunciations is also provided. Our goal is to add these new entries to the system as efficiently as possible, either directly before decoding or possibly even during decoding as part of a dialogue system. This is slightly different to previous work on Out-Of-Vocabulary (OOV) recognition where the lexicon is often augmented with sub-words and the OOVs are detected in another pass from either n-best lists or lattices [12, 11]

Recently, Allauzen et al [2] proposed lookahead composition that allows for the composition $det(L) \circ G$ to be performed very efficiently. Because G is composed during decoding one possible dynamic vocabulary approach would be to augment L with the new pronunciations, create the special lookahead type transducer, re-label the language model and compose C with L either statically or dynamically during decoding. However, the conversion time would introduce a latency that would be un-acceptable for many applications such as a dialogue system. Furthermore, the task of removing entries would also require the L transducer to be re-built and other transducers to be re-processed.

Next we describe the implementation of our fused composition algorithm, then we describe the extensions that allow OOV words to be added to class models on-the-fly. By OOV word we mean a new vocabulary item that does no occur in the pre-compiled L transducer but is specified separately before decoding.

2.1. Fused Composition Algorithm

A class n-gram model can be represented as two transducers, an n-gram model of class labels G, and a transducer T that maps from class labels to word labels. The expansion of G with T can be performed using standard composition and projection operations [1]:

$$GT = Sort_1(Proj_2(G \circ T)) \quad (2)$$

Here the subscripts 1 and 2 denote input or output labels respectively. $Sort_1$ sorts the arcs according to input labels, $Proj_2$ replaces all output labels in GT with the word ouput labels. The final $Sort_1$ is necessary to make the subsequent composition with CL more efficient. During speech decoding the entire cascade is:

$$CL \circ (Sort_1(Proj_2(G \circ T))) \quad (3)$$

Where CL is shorthand for $C \circ det(L)$. The static expansion of a class n-gram will often require substantial memory as each class label in G could potentially be multiplied by every transition in T. Therefore the expansion of GT is performed on-the-fly. To allow for memory efficient composition with CL we use OpenFsts [3] `CachedLogAccumulator` instead of the default `FastLogAccumulator`. The `FastLogAccumulator` will perform a traversal of all states in $G \circ T$ leading to the same memory explosion as the equivalent static composition. Whereas, the `CachedLogAccumulator` will only compute the lookahead scores for the states accessed during decoding.

Our proposed algorithm exploits the limited topology of the G and T WFSTs. We assume T has a single state and each arc represents a mapping from a class label to a single word label. Therefore the WFST resulting from the composition $G \circ T$ will have has the same number of states as G. Under these constraints the composition and sort is equivalent to merging a set of sorted lists, and it is a simple addition to also fuse the projection operation into the merging process. The entire algorithm is implemented as a single specialized class built around the OpenFst `CacheFst` type, this allows the specialized class to be seamlessly used as any other OpenFst class.

The fused GT WFST type aggregates G and T and works as follows: T is represented as a set of lists each sorted by word labels and indexed by the class labels. During decoding given a state s in GT, the first step is to iterate over the k arcs leaving state s in G and create a list of k class-to-word lists. A min-heap keyed on the output labels is used to merge the k lists with a total running time of $O(nlog(k))$ [14], where k is the total number of arcs leaving s and n is the sum of arcs across the k class lists. The fused algorithm also removes the memory requirement of the lazy project and sort operations.

2.2. Extension to Handle Dynamic Vocabulary

In our system all of the transducers except T are pre-compiled and once in the deployed system cannot be altered. The class model T is constructed by the decoder from a user supplied text file containing a set of class of word mapping with probabilities.

Often users would like to add words that do not occur in the pre-compiled lexicon L. In this case we require the user to supply additional pronunciation information such as a phoneme or Japanese Kana sequence. We place no restriction on the choice of units, however, we do require each unit occurs in the pre-compiled lexicon as a distinct vocabulary entry. For example if the units are phonemes each phoneme must occur as a word in the lexicon, this will ensure the $C \circ det(L)$ will be able to handle any dynamic cross word context dependencies correctly.

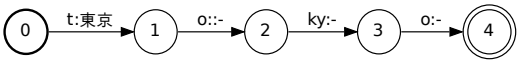


Figure 1: Example of an expanded OOV chain.

In our algorithm the CL and G transducers are totally untouched and instead we perform all the modification in the class-to-word mapping machine T. Lookahead composition [2] is an essential part of our algorithm, not only does it avoid the need to statically compose CL, G and T, it ensures the correct context dependency of dynamic vocabulary patched into T.

When our algorithm encounters an OOV entry in T it creates a linear chain of arcs that maps from the unit sequence to the new word ID. The chain transducer is a simple left-to-right transducer with an arc for each input unit, all the output labels are null except for the first transition which carries the OOV word label as shown in Figure 1. Based on the notation in [9] we call this operation $A = chain(P)$ where P is a unit to word sequence.

To avoid conflicts with the state numbering assumptions made in section 2.1 the chain expansion assigns new states IDs to the next free state ID value greater than $|G|$ and after the other allocated chain state IDs. This is operation is equivalent building the chain transducer and performing a general WFST *replace* operation. The replace operation substitutes an arc for an entire automaton, however the specialized approach has the advantage of avoiding the introduction of epsilon transitions that are artifacts of the OpenFst replace operation.

2.3. WFST Based Unit Selection

The dynamic vocabulary approach described in the previous section has the disadvantage in which the chains are built from sequences of short units. Such chains will not be represented as shared sequences in the final search space. We now describe a more efficient selection algorithm that matches the dynamic vocabulary to a more optimal set of existing lexicon items.

In the case of Japanese the OOVs are frequently proper nouns for example *Tokyokuogyodaigaku* (Tokyo Institute of Technology) is the concatenation of three nouns Tokyo, kuogyo and daigaku all of which occur in the lexicon. The idea behind our algorithm is to use the OOV pronunciation sequence to select a set of entries from the lexicon that matches the same overall OOV pronunciation whilst simultaneously minimizing the length of $|A|$. This will lead to better prefix sharing than naively expanding context dependant phone sequences. The proposed unit selection only considers the phonetic sequence and therefore is not restricted to linguist constraints, often totally unrelated words are selected based on the best phonetic match. We refer to the larger units as fragments as described in [12]. Fragments are units which have variable number of phonemes. In our system a fragment can be any unit which occurs in the static lexicon L and could range from a single phone, sub-word, word to even a composite word.

The selection process can be realized using a dynamic programming algorithm, here we wish to find the minimal number of fragments from the pronunciation lexicon that match the pronunciation of the OOV. There is an equivalence of dy-

namic programming and finding shortest (longest) paths in directed acyclic graphs, and this makes it possible to express the fragment selection using WFSTs. This allows a single generic WFST implementation to perform the unit fragmentation and depending on the selection criteria we can use heuristic or statistical driven methods such as a learnt grapheme-to-phoneme by simply changing the transducer.

The first stage is to build a pronunciation to word mapping transducer. WFSTs of this structure occur in many language processing applications, for example the lexicon transducer L used in an ASR cascade. In [4] such a transducer is presented for word normalization, the notation and description of our fragmenter is heavily by influenced by Chelba et al's [4] description.

Let P be a set of vocabulary entries comprised of pronunciation strings and the word identifiers taken from the from the recognition lexicon. We first construct a subset P' which contains only a single entry for each possible pronunciation realization. For each element p' of P' a phoneme to word transducer is constructed. We then construct the transducer M by taking the union of all of these chains:

$$M(P') = \bigoplus_{p' in P'} chain(p') \tag{4}$$

The next stage is to close and optionally optimize the transducer according to $M = det(M(P')^+)$, where $^+$ is the plus closure operations and det is the determinization operation. The final determinization does not affect the accuracy of the system but hugely improves the composition efficiency of M with other transducers. M is weighted according to the selection criteria, for example minimal number of fragment joins or greedy style longest fragment matches. Composing an OOV $chain(I)$ with M gives a lattice of all possible fragmentations, from which the best fragmentation can be extracted by finding the shortest path according to:

$$F = epsrm(shortestpath(M^{-1} \circ chain(I))) \tag{5}$$

Where $epsrm$ is the generic epsilon removal operation. The result of the above operation is a chain transducer representing a mapping from the *best* sequence of lexicon units to an OOV word ID. This chain can be used directly as the sequence expanded during speech decoding.

3. Experiments

In this section we evaluate our algorithm on a test-set used for NICT's *VoiceTra* [10] speech translation application. The VoiceTra service is a speech translation application designed to help foreign tourists in Japan. The test-set was comprised of 300 utterances which a contained a total of 182 OOV word instances not covered in the base lexicon and language model.

The baseline system we evaluated was for the Japanese language and built from the following models. A language model was trained using the data from the Basic Travel Expression Corpus (BTEC) corpus and the Hosei corpus, these are multilingual corpora of tourism style phrases. The training text consisted of 84k sentences which were annotated with a POS tagger, and seven proper noun classes were added to the system. In total the seven classes could be fully expanded into 18k words. The tagged text was used to build a Witten-Bell [16] smoothed language model using the SRILM toolkit [15]. The final language model had approximately 2.5M n-grams with a 48k vocabulary.

The acoustic models were a pair of gender dependent models each with 5690 states and 10 diagonal Gaussians per state. The acoustic models were trained on feature vectors comprised of 12 MFCCs with their deltas, augmented with a delta power term. To use these gender dependent models acoustic models we created a context-dependency transducers for each gender C_f and C_m and took the union according to:

$$min(det((C_f \cup C_m) \circ L)) \circ G \circ T \tag{6}$$

The composition and optimization of C with L is performed statically, that is $CL = min(det((C_f \cup C_m) \circ L))$. The class language model is composed dynamically during search by the decoder according to $CL \circ G \circ T$. The use of multiple acoustic models is similar to the approach proposed in [6]. However, when performing dynamic composition we find negligible memory increases when compared to a gender independent acoustic model. The worst case slowdown in decoding is around 10% which is substantially cheaper than running two gender dependent decoders.

The proposed algorithm was evaluated in NICTs *SprinTra* decoder [5], this is a general purpose one-pass Viterbi decoder. The memory usage was measured by overriding the global memory allocation functions and tracking the number of bytes requested by the decoder. The memory usage was recorded after processing each utterance and these values were averaged over the entire test-set. Any WFSTs that contained dynamically allocated transitions were completely destroyed and re-instantiated after processing 32 utterances to prevent memory usage growing too large.

To evaluate the proposed dynamic vocabulary scheme we compared three systems. A baseline system where we added the OOV words to the static lexicon L and built a standard CLG cascade. We considered two OOV systems where the missing vocabulary was specified as an additional set of class to word mappings. The first OOV system was a phone-based OOV (POOV) approach where each of the OOV chains were constructed directly from the phone sequence supplied in the OOV class mappings. The second OOV system was fragment-based OOV (FOOV) setup, we used the WFST algorithm proposed in section 2.3 to convert the OOVs to chains that contain the longest possible matches from the existing lexicon items.

Figure 2 shows the performance of the various approaches. The result shows that the FOOV system in comparison to the baseline converges slightly slower to the same asymptotic accuracy. This is due to the larger search space of the FOOV system and an increase in the number active state and arcs interacting differently with pruning. This fact is also reflected in the slightly higher memory usage in comparison to the baseline. The FOOV is substantially better than the POOV system which suffers low performance especially for narrower beams, the upper part of Figure 2 shows the expansion of longer phoneme sequences in the POOV is more costly in terms of memory.

For comparison a system built without any OOV support suffer a substantially reduction in accuracy, around 25% lower than the asymptotic best performance of the baseline system. This is due to large number of utterances with high OOV rates.

To highlight the space advantages of the fused algorithm we took the baseline system and compared the memory usage to the decoder using generic WFST operations to realize equation 3. On average the fused algorithm gave a 35% reduction in memory usage.

4. Conclusions

In this paper we have described an efficient approach for handling dynamic vocabulary in the WFST framework. The ex-

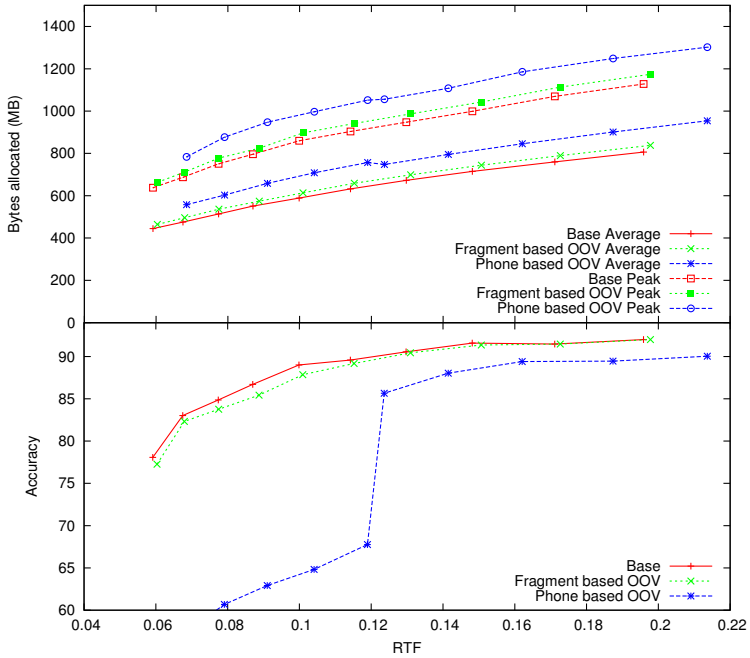


Figure 2: Comparison of different dynamic OOV approaches in comparison to a baseline with no OOVs on a voice search task. The upper plot shows the peak and average bytes allocated for each approach. The lower plot shows the accuracy for each of the approaches as function of real time factor (RTF).

periments have shown the specialized algorithm has a memory usage advantage over generic WFST operations. The proposed dynamic vocabulary approach allows for new words to be efficiently switched into the system without requiring changes to the lexicon or language model transducers. An efficient unit selection algorithm in the WFST framework is proposed, and the experimental results show there are significant performance gains when using larger lexical units in comparison to smaller phone units.

For future work we would like to evaluation larger and more complicated models, and extend the approach for dynamic situations that require changing vocabulary during decoding, such as part of a dialogue system or vocabulary adaption in a lecture transcription system with an information retrieval component. Further evaluations are needed to ascertain if the proposed approach will work equally as well on non-agglutinative languages. Other extensions to the technique could allow more arbitrary automata to be patched into the class models, such as regular expression for multiple pronunciations, G2P networks or even entire sub-grammars. It may be possible to achieve better fragment selection by performing a global fragment search and incorporate more knowledge from the task or language model.

5. References

[1] C. Allauzen, M. Mohri, and B. Roark, "Generalized algorithms for constructing statistical language models," in *Proc. of 41st Annual Meeting of the Assocication for Computational Linguistics*, 2003, pp. 40–47.

[2] C. Allauzen, M. Riley, and J. Schalkwyk, "A generalized composition algorithm for weighted finite-state transducers," in *Proc. Interspeech*, 2000, pp. 1203–1206.

[3] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Proc. of CIAA 2007*, 2007, pp. 11–23.

[4] C. Chelba, J. Schalkwyk, T. Brants, V. Ha, B. Harb, W. Neveitt, C. Parada, and P. Xu, "Query language modeling for voice search," in *Proc. IEEE Workshop on Spoken Language Technology*, 2010, pp. 127–132.

[5] P. R. Dixon, C. Hori, and H. Kashioka, "A comparison of dynamic WFST decoding approaches," in *Proc. ICASPP*, 2012, pp. 4209–4212.

[6] T. J. Hazen, I. L. Hetherington, and A. Park, "FST-Based Recognition Techniques for Multi-Lingual and Multi-Domain Spontaneous Speech," in *Proc. Eurospeech*, 2001, pp. 58–65.

[7] I. L. Hetherington, "A multi-pass, dynamic-vocabulary approach to real-time, large-vocabulary speech recognition," in *Proc. Interspeech*, 2005, pp. 545–548.

[8] M. Mohri, F. C. N. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," *Springer Handbook of Speech Processing*, pp. 1–31, 2008.

[9] M. Mohri, "Learning languages with rational kernels," 2009.

[10] NICT, "Voicetra," http://mastar.jp/translation/voicetra.html.

[11] C. Parada, M. Dredze, A. Sethy, and A. Rastrow, "Learning subword units for open vocabulary speech recognition," in *Proc. ACL*, 2011, pp. 712–721.

[12] A. Rastrow, A. Sethy, B. Ramabhadran, and F. Jelinek, "Towards using hybrid word and fragment units for vocabulary independent LVCSR systems," in *Proc. Interspeech*, 2009, pp. 1931–1934.

[13] J. Schalkwyk, I. L. Hetherington, and E. Story, "Speech recognition with dynamic grammars using finite-state transducers," in *Proc. Eurospeech*, 2003, pp. 1969–1972.

[14] S. S. Skiena, *The Algorithm Design Manual*, 2nd ed. Springer, 2008.

[15] A. Stolcke, "SRILM – an extensible language modeling toolkit," in *Proc. ICSLP*, 2002, pp. 901–904.

[16] I. Witten and T. Bell, "The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression," *IEEE Transactions on Information Theory*, vol. 37, no. 4, 1991.