A Multiplatform Speech Recognition Decoder Based on Weighted Finite-State Transducers

Emilian Stoimenov and Tanja Schultz

Cognitive Systems Labs, Institute for Anthropomatics, University of Karlsruhe
Am Fasanengarten 5, D-76131 Karlsruhe, Germany

{emilian, tanja}@ira.uka.de

***Abstract*— Speech recognition decoders based on static graphs have recently proven to significantly outperform the traditional approach of prefix tree expansion in terms of decoding speed [1], [2]. The reduced search effort makes static graph decoders an attractive alternative for tasks concerned with limited processing power or memory footprint on devices such as PDAs, internet tablets, and smart phones. In this paper we explore the benefits of decoding with an optimized speech recognition network over the fully task-optimized prefix-tree based decoder IBIS [3]. We designed and implemented a new decoder called SWIFT (Speedy WeIgthed Finite-state Transducer) based on WFSTs with its application to embedded platforms in mind. After describing the design, the network construction and storage process, we present evaluation results on a small task suitable for embedded applications, and on a large task, namely the European Parliament Plenary Sessions (EPPS) task from the TC-STAR project [20]. The SWIFT Decoder is up to 50% faster than IBIS on both tasks. In addition, SWIFT achieves significant memory consumption reductions obtained by our innovative network specific storage layout optimization.**

I. INTRODUCTION

Weighted finite-state transducers (WFSTs) [4] have been explored in various scenarios and on multiple tasks as a means of speech recognition search space construction. AT&T described their commercial speech recognizer based on WFSTs in [5], and IBM reported having written multiple WFST decoders [6]. In the WFST approach, a single unified network is constructed, in which the decoder searches for the path that corresponds to the most likely word string hypothesis given the spoken input, i.e. the sequence of observation vectors. This network comprises of the different knowledge sources used in speech recognition, approximated or represented exactly as transducer mappings, and then combined using weighted composition.

Less orthodox approaches have also been developed, such as exploring on-the-fly composition to balance between decoder run time performance and network size. In [7] a specialized composition algorithm has been described, and then some more general approaches followed in [8], [9], and [10].

There have been a number of developments for embedded platforms as well. The main implementation challenge for embedded platforms is the need for fixed-point arithmetic due to the lack of a floating point processor. In [11] a short text message dictation system is described, which uses a moderately sized language model, albeit a context independent acoustic model to keep the network size small. Earlier, [12] presented the application of codebook quantization and decoding with finite-state transducers with fixed-point weights.

Direct head-to-head comparison with the well-established prefix-tree expansion approach used in most commercial and community speech recognizers has not been as common. Kanthak *et. al.* [1] compared the AT&T recognizer with the one from RWTH, with results in favour of the WFST approach as implemented by AT&T. In a comparison from Philips [2], an unweighted decoding graph is successively optimized and the influence of various optimization techniques is explored. The author demonstrates how to apply the same optimizations to a prefix tree and obtains similar performance improvements with their dynamic decoder. Moore *et. al.* [13] conducted preliminary tests of the WFST decoder "Juicer" against the HTK decoder HVite, showing that at tighter beam widths Juicer is outperformed by HVite.

We were interested in developing our own WFST-based recognizer SWIFT and put it to the test against the contemporary prefix-tree based speech decoder IBIS [3] using a small task appropriate for embedded devices. This was done in order to assess the behaviour of WFST-based search under processing power and footprint constraints, as described in Section II.

To verify our implementation and to further study the SWIFT decoder, we also tested the decoder on a more demanding corpus taken from the 2006 TC-STAR evaluation. For evaluation, we converted models developed by University of Karlsruhe to a WFST network and compared the run time behaviour of both decoders. To allow fair testing, we integrated the SWIFT decoder into the Janus Recognition Toolkit and kept the acoustic and language models the same for both decoders. In these tests, the signal pre-processing and Gaussian evaluation code were shared to eliminate any impact of those factors to the comparison.

The following section II describes the experimental setup used to compare the decoders. Section IV gives an overview of the recognition network construction process and Section III describes how we applied it for both tasks. It also shows the sizes of the resulting networks. Section V outlines an improvement over existing network memory layout storage strategies, which turned out to significantly reduce memory consumption. The decoder implementation along with some improvements we discovered along the way of building a truly efficient search engine is explained in Section VI. Finally,

section VII presents the results of the experiments and concludes with a discussion and future work.

II. Evaluation Setup

The first evaluation was run on BTEC (Basic Travel Expression Corpus), a multilingual collection of conversational phrases in the travel domain developed by ATR [19]. The test set consists of 9 speakers and a total of 386 utterances. We used a language model with 8000 bigrams and 14996 trigrams trained on classes, representing street names, proper names, places of interest, and events. The total test vocabulary consisted of 3005 words. The acoustic model trained for this task was a set of three-state fully-continuous pentaphone HMMs with a total of 4000 Gaussians. A standard front-end was used, whereby 15 consecutive frames represented by the first 13 MFCCs were combined into a 195 dimensional vector, transformed by a Linear Discriminant Analysis matrix to produce a 32 dimensional observation vector. The asymptotic word accuracy of these acoustic and language models under wide beams is 81%.

To confirm the results from this first BTEC task, we also ran both decoders on the public condition of the 2006 TC-STAR evaluation. It consists of 1676 utterances, distributed over 51 speakers in sessions of the European Parliament [20]. The acoustic models were trained on approximately 80h of English EPPS data, 9.8h of TED data, and 167h of unsupervised EPPS training material. The unsupervised training material was obtained by adapting the acoustic model of a 2006 system on the output from RWTH Aachen and decoding the data, while using the segmentation provided by RWTH Aachen. We applied pentaphone semi-continuous models using 16,000 distributions over 4000 Gaussians with a variable number of Gaussians per model.

The language model is an interpolation of four 4-gram language models trained on 7 different sources with a total of 424,400,000, words. The final combination consists of ~10 M bigrams, 10M trigrams, and 11M fourgrams over a 40,000 word vocabulary.

The signal pre-processing of the EPPS system uses the first 13 MFCCs extracted from the FFT power spectrum. For the FFT a hamming window with a length of 16ms was applied using a window shift of 10ms. Fifteen adjacent frames were stacked into one feature vector and LDA was used to reduce the dimension of the feature vector to 42. The first speaker-adapted pass of the complete system achieves a word accuracy of 89.3%

Due to memory limitations, a recognition network based on the full fourgram language model of the EPPS system could not be built. Therefore we chose to prune it using the SRI LM toolkit [21] with a threshold of 1e-7. This resulted in a language model with about 1 M bigrams, 342,000 trigrams, and 80,000 fourgrams over the same 40,000 words. The accuracy of both decoders using this language model is 87.7%.

III. WFST-based Decoding

As originally proposed by Mohri *et. al.* [4] a *weighted finite-state transducer* (WFST) that translates phone sequences into word sequences can be obtained by forming the *composition* $L \circ G$, where L is a *lexicon* which translates the phonetic transcription of a word to the word itself, and G is a *grammar* or *language model* which assigns to valid sequences of words a weight consisting of the negative log probability of this sequence. In the original formulation of Mohri and Riley [22], phonetic context is modelled by the series of compositions $H \circ C \circ L \circ G$, where H is a transducer converting sequences of Gaussian mixture models to sequences of polyphones, and C is a transducer that converts these polyphone sequences to corresponding sequences of phones. An alternative more efficient approach which reduces the computational effort of this explicit phone context expansion was proposed in [23] and corrected in [15]. This method constructs the composed transducer $H \circ C$ directly by enumerating the possible HMM sequences and connecting them accordingly. We call this transducer *'HC'* to reflect the difference with the explicit composition of H and C.

The integrated transducer $HC \circ L \circ G$ contains many redundant paths, which can be merged together by means of *weighted determinization* and *weighted minimization* [4]. These algorithms are generalizations of the corresponding algorithms for unweighted automata. When applied together to the final recognition transducer, they have the effect of merging together the equivalent GMM sequence expansions of word prefixes and suffixes.

An integral part of weighted minimization is *weight pushing*. It moves the weight along a path of the transducer as near as possible to the initial node. In addition to enabling a more efficient minimization, weight pushing also makes pruning more effective, because the bulk of the path weight is placed in its beginning and the decoder can discard unpromising paths as early as possible. However, Mohri *et. al.* [4] noted that pushing using the max operation of the *tropical* semi-ring for combining the weights of the paths from a node actually reduces the speed of the decoder. We observe the same performance degradation in our experiments.

Figure 1 shows a typical path of the determinized and minimized integrated transducer $HC \circ L \circ G$ constructed for the TC-STAR task as described in the following section.

IV. Network Construction

A. Phone level transducers

The experiment on the small BTEC task uses a class based language model. Representing class based language models with WFSTs has been already described in [14], but we include it here for sake of completeness. Assuming non-overlapping word classes, a trigram model assigns

$$P(W) = \prod_i P\left(w_i \middle| c(w_i)\right) P\left(c(w_i) \middle| c(w_{i-2}), c(w_{i-1})\right)$$

to a sequence of words $W = (w_1, w_2, \ldots, w_n)$, where $(w_i, c(w_i))$ is the word-class relation. This probability can readily be represented by the composition of the language model transducer G with a class transducer K in either the tropical or the log semi-ring. The structure of K is a simple loop,

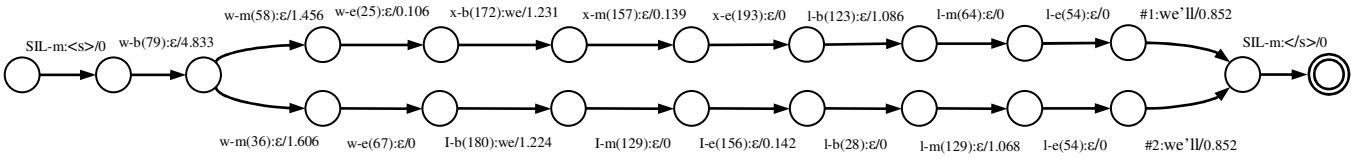

Figure 1. Two paths from the TC-STAR network showing the GMM expansion of the word trigram "<s> we'll </s>". The two paths correspond to the two pronunciation variants of "we'll" : "w x l" and "w I l". The label notation is input:output/weight. The sum of the weights along both paths is the same and is given by the language model. The symbols "#1" and "#2" are the disambiguation symbols added to be able to determinize $L \circ G$ as described in [4].

mapping classes to their word members with the corresponding class conditional probability. However, the composed transducer $G \circ K$ is non-functional because of the many-to-one nature of the word-class mapping. Therefore, it needs to be *projected* on the output side and then determinized. A consequent straightforward composition with a dictionary transducer L followed by determinization, weight pushing, and minimization results in an optimal transducer mapping phone units to words, together with the correct class history probability.

B. Acoustic model transducers

The acoustic model in both tasks uses pentaphones. To model the context dependencies, we converted the decision tree to a directly composed '*HC*' transducer using the method described in Section III. The size of the *HC* transducer built for the EPPS task was too large to be composed into a network, much less for the resulting network to be determinized. To overcome this problem, we made use of the fact that the 16,000 different distributions on the leaves of the decision tree are mapped to only 4,000 GMMs in the semi-continuous model. We replaced the distribution symbols with the corresponding GMM symbols, then determinized and minimized the transducer to obtain a new transducer HC_{GMM}. The size of the transducer shrank by a factor of almost nine which allowed us to build an integrated recognition network. Table I shows the sizes of the optimized combined *HC* transducers.

TABLE I
SIZES OF THE COMBINED HC TRANSDUCER FOR BOTH TASKS.

	Nodes	Arcs
BTEC *HC*	20,263	215,189
TC-STAR *HC*	1,456,145	9,774,218
TC-STAR HC_{GMM}	157,261	1,127,032

The sizes of the optimized $H C \circ L \circ G$ transducers for both tasks are shown in Table II.

TABLE II
RECOGNITION TRANSDUCER SIZES FOR THE BTEC AND THE TC-STAR TASKS.

	Nodes	Arcs
BTEC	1,633,975	2,178,331
TC-STAR	13,755,867	21,409,106

All network composition and optimization operations were performed with OpenFst [18].

V. NETWORK LAYOUT

The most common network layout implementation is a linear array of arc structures, sorted by the source node, with fields indicating the input GMM index, the output index, and the weight. A couple of improvements can be made on this network organisation to save space. Caseiro [7] employed a variable bit encoding that uses the properties of the network to reduce the average number of bits per state. Saon *et. al.* [6] and Olsen *et. al.* [11] described a similar network arrangement. The benefits of the network layout in [7] are negated by the memory alignment properties of most modern computing platforms, therefore Caseiro suggests accessing the edges in chunks during the decoding. This requires a new index layer to map from states to chunks.

In contrast, we implemented a simpler approach which builds upon the fact that the network is not factored. Combined with the transducer / acceptor conversion in [6], this approach allows us to compress the small BTEC network to about half of its size on disk when encoded using 10 Bytes per arc.

The *HC* transducer expands every context dependent triphone to a sequence of GMM indices. Many of those three state sequences are unique and produce linear arc chains in the transducer. Others are not unique, but participate in word phone expansions which could not be merged during the determinization and minimization of $L \circ G$. In [4], these chains are compacted into one arc by means of a second transducer mapping chains to HMM specification symbols. This way, only the compacted transducer is used in the decoding, the HMM specification symbols being expanded by the decoder on demand as part of the HMM evaluation routines.

We do not have a variable length specification for the HMMs as in [4], but we still would like to reduce the impact of the linear arcs on the transducer size. Replacing the linear succession of arcs to one arc and expanding it on-the-fly would complicate the token management in the decoder. The alternative is to decrease the amount of memory required by an arc in a linear sequence. Indeed, the indices that label the states along the chains can be chosen in linear succession and only the beginning state index from the branch can be stored. A single bit taken from the symbol label can mark an arc involved in such a linear chain. When propagating a token, the transducer access program code knows the node index that contains the token, and calculates the new target node index

by incrementing the old one if it is propagating the token along a chain. This way the information necessary to store this arc becomes the triple (*input, output, weight*). Furthermore, weight pushing moves the weights on the branches of the transducer, leaving zero weight along the linear chains. Thus, the only information needed to store for an arc in a chain is the input/output pair. Moving the output side to the input side removes the necessity to store the transducer output alphabet, but can increase the size of the transducer because new arcs have to be added where both the input and the output side are different from epsilon. In our experiments, these additional arcs were very few as shown in Table III and made the conversion worthwhile.

TABLE III
RELATIVE TRANSDUCER SIZE INCREASE WHEN CONVERTED TO AN ACCEPTOR (COMPARED TO THE ORIGINAL TRANSDUCER SIZES IN TABLE II).

	Additional Arcs	Size Increase
BTEC	104,761	4.8%
TC-STAR	2,427,719	11%

A depth first search writing the indices of the graph nodes as they are discovered would produce the desired state arrangement. Unfortunately, it has the potential to break a useful property of the decoder: the fact that it accesses the network in a breadth first manner. This would deteriorate the cache locality and possibly slow down the decoder. Therefore, an algorithm for arranging the nodes which performs breadth first search and switches to depth first search on the linear paths is a suitable alternative to get the best of both worlds.

We have to note that this storage scheme can be applied without modification to the branches in the network by numbering all target nodes at a given branch with consecutive numbers. Unfortunately, this is not as efficient, because not all branch target nodes can be renumbered this way due to the cyclic nature of the network. Also, not all target nodes in a chain path can be consecutively numbered, namely the target node of the last chain has to retain its original index, because the in-degree of the last node is greater than one. This gives rise to the following categorization of all transducer arcs:

a) branch arcs which can be consecutively numbered; [ConsBrnchArc]
b) branch arcs which lead backwards into the network and cannot be consecutively numbered; [BrnchArc]
c) chain arcs; [ChnArc]
d) arcs that are last in a chain. [LstChnArc]

The following tables show the percentage of arcs types contained in each of the networks we examined.

TABLE III
DISTRIBUTION OF THE ARC TYPES IN THE BTEC NETWORK

	Arcs	Percentage
CnsBrnchArc	237,602	10.40 %
BrnchArc	558,237	24.45 %
ChnArc	1,203,028	52.69 %
LstChnArc	284,226	12.45 %

TABLE IV
DISTRIBUTION OF THE ARC TYPES IN THE TC-STAR NETWORK

	Arcs	Percentage
CnsBrnchArc	4,246,202	17.81 %
BrnchArc	11,940,612	50.09 %
ChnArc	6,763,075	28.37 %
LstChnArc	886,933	3.72 %

ChnArcs amount to more than half of the BTEC network and more than one fourth of the TC-STAR network. The branch arcs in the TC-STAR network are more than the chain arcs, because of the high degree of sharing between the three-state HMM sequences in the TC-STAR *HC* transducer. It would be desirable to apply the compression to both CnsBrnchArcs and ChnArcs at the same time; however, the simultaneous renumbering of both arc types is impossible in general. The reason is that chains can directly follow a branch, and thus break the consecutive ordering of the branch target nodes as shown in Figure 2.

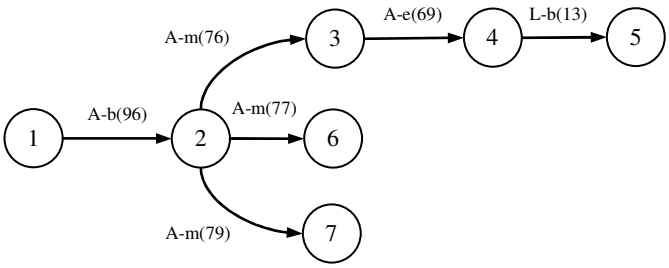

Figure 2. A branch in the network, followed by a chain. The arcs are labeled by GMM names. The chain {A-e(69), L-b(13)} follows a branch at node 2. Node 6 cannot be numbered 4, because the index is used by the target node of arc A-e(69).

Table V shows the network sizes in Megabytes on the BTEC and TC-STAR experiments when saving only the input label for ChnArcs and the (*target, label, weight)* triple for all other arc types. A comparison is given with the memory usage of the recognition *transducers*, which are binary encoded with 10 bytes per arc (4 bytes for the target node index, two 2 byte label fields and another 2 bytes for the weight).

We compressed the BTEC and TC-STAR networks using 2 bytes for the label of all ChnArcs. For all other arc types, we needed 4 Bytes for the target node index, 2 bytes for the label and 2 bytes for the weight. This encoding results in a compression rate of 4.8 bytes per arc for the small network and 6.3 bytes per arc for the large one. This method achieves higher efficiency than the one in [7] where the best storage rate was 5.2 bytes per arc on a recognition transducer with context independent phones and 7.2 bytes when encoding language model transducers.

TABLE V
MEMORY USAGE OF THE COMPRESSED NETWORKS

	Standard	ChnArc
BTEC	20.7 MB	10.5 MB
TC-STAR	227.3 MB	143.16 MB

The only function of the nodes in the transducer network is to link and combine alternative paths, because the observation distributions are put on the input side of the arcs and many arcs with different input symbols can end in a node. That is why after renumbering the network the node indices can be completely replaced by offsets in the array of arcs. This way, the setup becomes a permanently stored adjacency list graph representation, and the arc array is the only remaining structure that fully describes the transducer.

VI. Token Passing

Since the entire network construction work is offloaded in a pre-processing stage, the decoder can be written in a breadth-first search style as a token passing algorithm [16]. The decoder implementation keeps two active token queues: the queue q_{cur} processed in the current frame and the queue q_{next} which holds the propagated tokens from q_{cur}. The current frame queue is visited in a linear fashion and for each child node the language model score is added to the token score. If the score falls within a beam width distance from the current best score, the GMM on the input side of the arc is immediately evaluated and accumulated in the token. The token is then queued for propagation in the next frame by placing it into q_{next}. This happens unless the arc target node has not been already occupied by a token. To verify this, we use a hash table mapping static indices to indices in q_{next} as described in Saon *et. al.* [6]. In case the state has been occupied, and the score of the old token is worse, it is replaced by the new token.

At each new frame, the roles of q_{cur} and q_{next} are swapped by exchanging their pointers and q_{next} has to be cleared. This cleanup operation is expensive as the queue can hold tens of thousands of tokens which have to be invalidated at each new frame. However, the cleanup work can be avoided by storing an additional frame index in the token queue, which indicates *at which frame* the token was placed in the queue. This way, the hash table lookup described in the previous paragraph is complemented by an inexpensive check to see if the token found in the target state was placed there earlier during the processing of the *same* observation frame.

The decoder computes the acoustic scores as they are requested during the token propagation, avoiding duplicate acoustic score computation by storing the already computed scores in a cache. This is different from the prefix tree expansion and acoustic score computation implemented in IBIS. The IBIS decoder processes the frame in two steps: an expansion step, consisting of network expansion and token propagation, and a score computation and token pruning step. In the first step, the tokens are propagated across word and phone boundaries, and the relevant beams are applied. At the same time, the acoustic scores are *requested* for computation.

The second step computes the requested scores and applies them to the tokens, involving a second pass over the active tokens. We implemented this two-step method in the SWIFT decoder and compared the speed of both strategies. The results showed that the after-frame score evaluation is slower by relatively 5-10%. Nevertheless, the SWIFT decoder using the late evaluation scheme is still faster than IBIS.

VII. Experiments and Recognition Results

We started by tuning the IBIS decoder on the small BTEC task. After finding the best combination of language model weight and word insertion penalty on the development set, we proceeded to determine the relationship between Real Time Factor (RTF) and Accuracy (WA) shown on Figure 3. We then did the same for the SWIFT decoder. It can be seen from Figure 2 that even though the results are very similar, the SWIFT decoder converges faster to the optimal recognition rate. For example, at 79.5% WA, the SWIFT Decoder is 50% faster than IBIS. These experiments were performed on an Intel(R) Core(TM) 2 Duo CPU with each core running at 2.33GHz. IBIS had a peak resident memory usage at 53 MB, while the SWIFT Decoder's memory footprint was 52 MB.

We also performed different experiments with the SWIFT decoder to study the impact of pushing in the log and in the tropical semi-ring. The results show that the network constructed in the tropical semi-ring significantly slows down the decoder even given the small BTEC task. A plausible explanation for this is the Viterbi maximum approximation used when calculating the node potentials during weight pushing. The log semi-ring would sum the weights

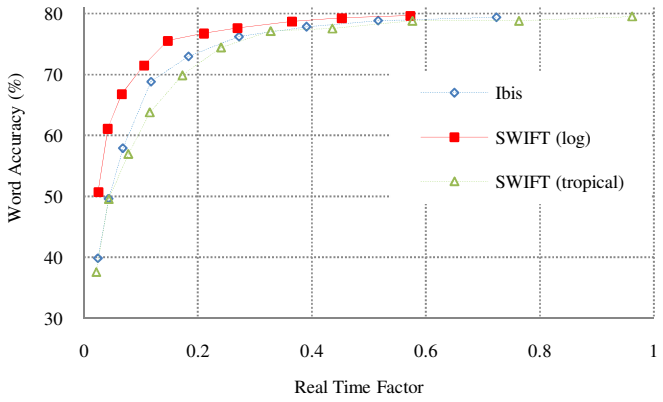

Figure 3. Comparison of SWIFT with the prefix-tree decoder Ibis on the BTEC corpus. We also show comparison of two networks whose weights are pushed in the log and in the tropical semi-ring respectively.

when combining the paths, and this way correctly implement the "or" semantics for the path probabilities given by the language model. Furthermore, the node potentials will be higher than those when using the max approximation. This results in more weight being pushed to the beginning of the network.

Figure 4 shows the corresponding WA/RTF relationship for the large vocabulary EPPS task. There we see results consistent with the BTEC task, although the improvement is more moderate. For example at 76.5% WA, the relative speed improvement is close to 26%. The difference between the decoders disappears at real-time factors larger than 3.5.

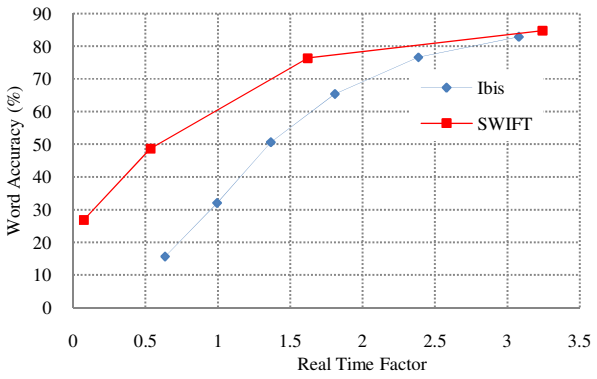

Figure 4. Comparison of the IBIS prefix-tree decoder and the SWIFT decoder on the large vocabulary EPPS task.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper we described the design and implementation of a WFST based decoder called SWIFT (Speedy WeIgthed Finite-state Transducer). The decoder is supposed to support multiple platforms, i.e. its design and implementation is meant to work out on PCs as well as on resource-constrained devices with limits in both, processing power and memory. We described new solutions to drastically reduce the network size by combining the factoring idea from [4] with a histogram analysis on the arcs in the network to construct a new network layout which does not hamper recognition speed and is simple to implement and use. Furthermore, we achieved a significant memory consumption reductions obtained by an innovative network specific storage layout optimization.

The resulting decoder was evaluated on two very different setups, a small vocabulary domain restricted conversational travel phrases task (BTEC) and a large vocabulary task on European Parliament Speeches (EEPS). The results of the SWIFT decoder were also compared to the JRTk IBIS decoder. In both tasks SWIFT achieved an up to 50% speed-up over IBIS.

Overall, we see significant improvements in recognizing both the relatively small BTEC task and the EPPS task with large acoustic and language models. The EPPS experiments showed that large tasks are still quite challenging, mostly due to the memory requirements for the determinization in the final network optimization step. Therefore, in the near future we plan to focus on adding dynamic composition to the decoder and optimizing the determinization in the final processing step. The arc compression does not pose a problem for the dynamic composition, because, similar to the decoder, it accesses the arcs of the transducers being composed in a breadth first search style. However, BrnchArcs might prove to be more suitable for compression in long span language models, because of the high average forward branching factor of their transducer approximations.

ACKNOWLEDGEMENTS

The authors wish to thank Thilo Köhler and Christian Fügen for providing the BTEC system and their active support. We are also grateful to Sebastian Stüker for making the speaker adapted first pass of University of Karlsruhe's submission to the 2007 TC-STAR evaluation available to us.

REFERENCES

[1] S. Kanthak, H. Ney, M. Riley and M. Mohri. "A comparison of two LVR search optimizations techniques." *ICSLP'02*, Denver, Colorado, USA, 2002.

[2] H. Dolfing. "A comparison of prefix tree and finite-state transducer search space modelings for large-vocabulary speech recognition," *ICSLP'02*, Denver, Colorado, USA, 2002.

[3] H. Soltau, F. Metze, C. Fügen, and A. Waibel. "A One Pass-Decoder Based on Polymorphic Linguistic Context Assignment", *ASRU*, Trento, Italy, 2001.

[4] M. Mohri, F. Pereira, and M. Riley. "Weighted finite-state transducers in speech recognition", *Computer Speech and Language*, 16:69–88, 2002.

[5] V. Goffin, C. Allauzen, E. Bocchieri, D. Hakkani-Tür, A. Ljolje, S. Parthasarathy, M. Rahim, G. Riccardi and M. Saraclar. "The AT&T Watson Speech Recognizer", *Interspeech '05*, Lisbon, Portugal, 2005.

[6] G. Saon, D. Povey, and G. Zweig. "Anatomy of an extremely fast LVCSR decoder", *Interspeech '05*, Lisbon, Portugal, 2005.

[7] D. Caseiro and I. Trancoso. "Using Dynamic WFST Composition for Recognizing Broadcast News", *ICSLP'02,* Denver, Colorado, USA, 2002.

[8] T. Hori, A. Nakamura."Generalized Fast On-the-fly Composition Algorithm for WFST-Based Speech Recognition", *Interspeech '05*, Lisbon, Portugal, 2005.

[9] O. Cheng, J. Dines, M. M. Doss. "A Generalized Dynamic Composition Algorithm of Weighted Finite-State Transducers for Large Vocabulary Speech Recognition", *ICASSP '07,* Honolulu, Hawaii, USA, 2007.

[10] J. McDonough, E. Stoimenov, D. Klakow. "An Algorithm for Fast Composition of Weighted Finite-State Transducers", *ASRU '07,* Kyoto, Japan, 2007.

[11] J. Olsen, Y. Gao, G. Ding, X. Yang. "A Decoder for Large Vocabulary Continuous Short Message Dictation on Embedded Devices", *ICASSP '08*, Las Vegas, Nevada, USA, 2008.

[12] E. Bocchieri, D. Blewett. "A Decoder for LVCSR Based on Fixed-Point Arithmetic", *ICASSP*, Toulouse, France, 2006.

[13] D. Moore, J. Dines, M. M. Doss, J. Vepa, O. Cheng, T. Hain. "Juicer: A Weighted Finite-State Transducer Speech Decoder", *MLMI '06*, Washington DC, USA, 2006.

[14] C. Allauzen, M. Mohri, B. Roark, and M. Riley. "A Generalized Construction of Integrated Speech Recognition Transducers", *ICASSP '04,* Montréal, Canada, 2004.

[15] E. Stoimenov, J. McDonough. "Memory Efficient Modeling of Polyphone Context with Weighted Finite-State Transducers", *Interspeech '07*, Antwerp, Belgium, 2007.

[16] S.J. Young, N.H. Russell , J.H.S. Thornton. "Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems", *Technical Report, University of Cambridge,* 1989.

[17] T. Köhler, C. Fügen, S. Stüker, and A. Waibel, "Rapid Porting of ASR Systems to Mobile Devices", *Interspeech '05*, Lisboa, Portugal, 2005.

[18] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, M. Mohri. "OpenFst: A General and Efficient Weighted Finite-State Transducer Library", *CIAA '07*, Prague, Czech Republic, 2007.

[19] G. Kikui and E. Sumita and T. Takezawa and S. Yamamoto. "Creating corpora for speech-to-speech translation", *Interspeech '03*, pp. 381-384, Geneva, Switzerland, 2003.

[20] Technology and Corpora for Speech to Speech Translation (TC-STAR). Integrated Project funded by the European Commission, Project No. FP6-506738, 2004-2007. http://www.tc-star.org.

[21] A. Stolke. "SRILM – An Extensible Language Modeling Toolkit", *ICSLP '02,* Denver, Colorado, USA, 2002

[22] M. Mohri and M. Riley. "Network optimizations for large vocabulary speech recognition". *Computer Speech and Language,* 16:69-88, 2002

[23] M. Schuster and T. Hori. "Efficient Generation of High-order Context-Dependent Weighted Finite-State Transducers for Speech Recognition", *ICASSP '05*, Philadelphia, PA, USA, 2005