論文 / 著書情報
Article / Book Information

Title	An Empirical Comparison of the T3, Juicer, HDecode and Sphinx3 Decoders
Authors	Josef R. Novak, Paul R. Dixon, Sadaoki Furui
Citation	INTERSPEECH2010, , , pp. 1890-1893,
Pub. date	2010, 9
Copyright	(c) 2010 International Speech Communication Association, ISCA
DOI	http://dx.doi.org/

An Empirical Comparison of the T^3, Juicer, HDecode and Sphinx3 Decoders

Josef R. Novak[1], *Paul R. Dixon*[2], *Sadaoki Furui*[1]

[1]Department of Computer Science, Tokyo Institute of Technology, Japan
[2]National Institute of Information and Communications Technology, Kyoto, Japan
novakj@furui.cs.titech.ac.jp, dixonp@furui.cs.titech.ac.jp, furui@furui.cs.titech.ac.jp

Abstract

In this paper we perform a cross-comparison of the T^3 WFST decoder against three different speech recognition decoders on three separate tasks of variable difficulty. We show that the T^3 decoder performs favorably against several established veterans in the field, including the Juicer WFST decoder, Sphinx3, and HDecode in terms of RTF versus Word Accuracy. In addition to comparing decoder performance, we evaluate both Sphinx and HTK acoustic models on a common footing inside T^3, and show that the speed benefits that typically accompany the WFST approach increase with the size of the vocabulary and other input knowledge sources. In the case of T^3, we also show that GPU acceleration can significantly extend these gains.

Index Terms: WFST, LVCSR, acoustic models, HTK, Sphinx

1. Introduction

Developing high quality baseline systems for Automatic Speech Recognition (ASR) evaluations, comparisons and new experiments often represents a challenging prospect. Research in this area enjoys a large number of options in terms of accessible ASR decoders, as well as acoustic and language model training frameworks. Yet without comparable baselines it can sometimes be difficult to analyze or interpret new results effectively.

In this paper we expand upon recent results from [1] and investigate the relative performance of the T^3 decoder [2] on two widely used Large Vocabulary Continuous Speech Recognition (LVCSR) evaluation sets based on the Wall Street Journal corpus (WSJ) [3]. We make an empirical comparison with the HTK HDecode [4], Sphinx3 and Juicer [5] decoders. We compare the T^3 decoder against previously reported baselines, utilizing the freely available WSJ acoustic models [6] and two standard WSJ languages models. Through the use of T^3 as a common decoding engine we are also able to evaluate HTK and Sphinx acoustic models on a common footing.

The remainder of the paper is structured as follows: Section 2 describes the shared knowledge sources employed for these evaluations while Section 3 describes the HTK and Sphinx models, Section 4 describes the Weighted Finite-State Transducer (WFST) cascade configuration, Section 5 outlines the experimental setup, Section 6 provides results and related discussion, and Section 7 concludes the paper.

2. Shared Knowledge Sources

In order to minimize workload and help guarantee the repeatability of our experiments, throughout this work we employ the freely available, pre-tuned acoustic models described in [6], and for two of three tasks rely on standard WSJ language models and pronunciation dictionaries. The third LVCSR task also relies on a freely available ARPA format language model.

3. HTK, Sphinx3 and Juicer

HTK and Sphinx are two of the more widely used open source speech recognition toolkits, and both have enjoyed considerable popularity over the years. More recently the Juicer WFST decoder has begun to gain traction as a popular WFST-based alternative to the tree-based dynamic decoders provided with the HTK and Sphinx toolkits. This section discusses some of the salient features of these systems, focusing on acoustic models and decoder characteristics.

3.1. HTK and Sphinx Acoustic Models

In both the case of Sphinx and HTK, the acoustic models were trained using the full set of WSJ0 and WSJ1 training data consisting of approximately 211 hours of data. This included the long-term and journalist training data. In the case of T^3, in addition to the existing HTK conversion tool, a new tool was developed to convert arbitrary Sphinx format acoustic models into a format suitable for use with the T^3 decoder.

Both the HTK and the Sphinx projects provide suites of tools suitable for training, updating, adapting and performing research on acoustic models, and while the resulting models and overall training procedures are in general quite similar, there are yet some differences. These differences, most of which were originally described in [7], pertain largely to implementation details rather than fundamental differences in approach and are summarized below,

- HTK applies sine-curve liftering to the output MFCCs.
- HTK does not scale the mel filters to have constant area.
- HTK uses a "textbook" DCT-2, but with a normalization term of $\sqrt{\frac{2}{N}}$ rather than $\frac{2}{N}$.
- The HTK recipe uses a very different set of filter bands from the standard Sphinx wide-band configuration.
- HTK does not round the filter edges to fall directly on DFT points.
- HTK supports training silence models with an independently variable number of Gaussians.

As described in [6], the freely available acoustic models which were employed for these experiments were trained so as to produce results that are, as far as possible, comparable in terms of their salient characteristics. Thus, both the HTK and Sphinx models employed roughly 8000 tied states, and 32 Gaussian mixture models. Further details of the acoustic models employed throughout this work are described in Table 1.

3.2. Decoders

The HTK and Sphinx projects each provide several well-known decoders suitable for use in LVCSR tasks. HTK provides HVite,

Table 1: Acoustic Model Characteristics

Model	*Topology*	*Tied States*	*Gaussians*	*Size*
HTK	3s no-skip	8000(apprx)	32	73MB
Sphinx	3s no-skip	8000	32	72MB

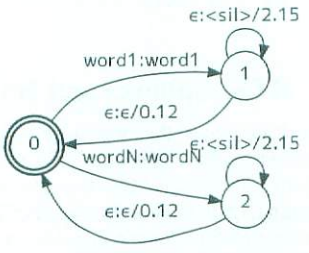

Figure 1: Silence-class Model T

as well as the more recent HDecode. Sphinx provides Sphinx2, Sphinx3, PocketSphinx, and Sphinx4. For these evaluations we chose to focus on HDecode and Sphinx3 which are arguably the most well-known and widely used among the above options.

More recently the Juicer [5] WFST decoder has been released as an open source project maintained by IDIAP. Juicer provides similar functionality to the T^3 WFST decoder in terms of the model inputs it accepts; it is capable of performing decoding on both static cascades, as well as on-the-fly composition, and it has been developed to read in HTK-based acoustic models in native format.

4. WFST Setup

The WFST cascades for the HTK and Sphinx acoustic models were both constructed using the same input knowledge sources that were used as input to HDecode and Sphinx3. Each of these individual knowledge sources was compiled into a WFST, where G represents the language model, L the lexicon, and C the context-dependency. Furthermore, the same HTK-based WFST cascade was used exactly as-is in both T^3 and Juicer.

A generic conversion program was written to transform ARPA format language models to WFSTs and this was employed to convert each of the language models into an equivalent WFST according to the strategy outlined in [8].

The lexicon and context-dependency transducers were also constructed using standard topologies, as described in [9]. Silence modeling was implemented through the use of a silence-class transducer, T, which was constructed following the topology outlined in [9], and is illustrated in Figure 1. Arc weights for the sil loop and exit arcs were estimated based on 400+ hours of force-aligned transcripts from the Fisher corpus [10].

The T transducer was composed with G prior to performing further downstream composition or optimization operations and the resulting $G \circ T$ cascade for each task was shared across all HTK and Sphinx based recognition networks.

Based on previous results from [9], as well as our own recent experiments in [11] all compilation, composition and optimization operations were conducted in the log semiring. The complete, exact series of operations is thus described below,

$$\pi(C \circ det(L \circ (G \circ T))) .$$

Here $\circ$ denotes the composition operation, determinization is denoted by det, and π represents the auxiliary symbol replacement operation.

Table 2: WSJ-based Cascade Characteristics

Cascade	*Arcs*	*States*	*Size*
nov92-5k-sphinx	2575545	721451	48MB
nov92-5k-htk	2529014	770393	48MB
si_dt_s2-20k-sphinx	4818666	1517563	91MB
si_dt_s2-20k-htk	4777668	1620681	91MB

Table 3: Large Cascade Characteristics

Cascade	*Arcs*	*States*	*Size*
CSR-64k-sphinx	171557031	114642583	3.9GB
CSR-64k-htk	155932098	99801462	3.5GB

Using the above construction recipe, HTK-based and Sphinx-based cascades were constructed for three different tasks of increasing difficulty, which are described in Section 5. In practice, and for the purpose of ensuring the greatest degree of overall equivalence, the HTK-based cascades were used without modification in both Juicer and T^3. Similarly, the $G \circ T$ component which represents the combination of the sil-class WFST and the language model, may be shared without alteration across HTK-based T^3, Juicer, and Sphinx-based T^3 cascades, and in practice this is exactly the way in which the cascades were compiled.

5. Experimental Setup

In these evaluations we look at three different LVCSR tasks based on two different well-established test sets of increasing difficulty. The first task, *nov92-5k*, focuses on the November 1992 ARPA WSJ test set which comprises 330 sentences, and was evaluated using the WSJ 5k non-verbalized vocabulary and the standard WSJ 5k closed bigram language model. The second task, *si_dt_s2-20k*, focuses on a subset of the WSJ1 Hub2 test set which comprises 207 sentences. The *si_dt_s2-20k* task, which is somewhat more difficult, was evaluated with the standard WSJ 20k non-verbalized closed bigram language model and corresponding vocabulary.

Network characteristics of the resulting WFST cascades for the *nov92-5k* and *si_dt_s2-20k* tasks are described in Table 2.

Table 4: T^3 Parameter Settings

Parameter	*Value*
Beam	90∼380
Band	10000
LM weight	15.0
Word insertion penalty	20.0

The third task, *si_dt_s2-64k* also focused on the more difficult si_dt_s2 test set, but looked at a much larger vocabulary and more complex language model. The model evaluated for this task covered a 64k vocabulary and included a 3-gram LM trained on approximately 222M words from the CSR LM-1 corpus [12], which comprised 13628086 bigrams and 8811112 trigrams. Along with the corresponding vocabularies and pronunciation dictionaries, which are suitable for use in HTK and Sphinx, this LM is also freely available for research purposes and can at the time of writing be found at the location described in [6]. Details of the two WFST cascades constructed for this task are described in Table 3.

The LVCSR evaluations for the *nov92-5k* task and the *si_dt_s2-20k* task were both run on an Intel Core 2 based machine running at 3GHz with 6MB of cache and 4GBs of main system memory. Due to significantly greater memory requirements, particularly where the WFST decoders were concerned, the evaluations for the *si_dt_s2-64k* task were run on an 8 core Intel Xeon based machine also running at 3GHz with a 6MB cache and 64GBs of main system memory. The platform in both cases was a 64bit Linux machine, the former running Fedora Core and the latter RHEL.

In order to obtain accurate Real-Time Factor (RTF) versus Word Accuracy curves, each decoder was evaluated on a wide range of different beam widths, while auxiliary parameters such as insertion penalty, language model weight, etc. were held constant.

In the case of T^3 these auxiliary parameters, which included the band, language model weight and insertion penalty were set to 10000, 15 and 20 respectively following initial manual tuning evaluations on a held out test set. Similar tuning evaluations combined with information from [13] were used to peg these auxiliary values for the Juicer WFST decoder. The baseline parameter values for T^3 are described in Table 4 while those for Juicer are described in Table 5.

In the case of HDecode and Sphinx3, which provide support for an exceptionally large number of auxiliary parameters, values aside from the main beam width were implemented exactly as recommended in [6].

Table 5: Juicer Parameter Settings

Parameter	*Value*
mainBeam	90~200
lmScaleFactor	15
insPenalty	-5
threading	Yes

For the *nov92-5k* and *si_dt_s2-20k* tasks, each of the WFST networks was evaluated in the T^3 decoder in two different modes. The first mode consisted in performing all computations on the CPU using only a single thread, and is referred to in figures as T^3-*-*sse*. The second mode leveraged T^3's native Graphics Processing Unit (GPU) support to perform acoustic likelihood computations on the GPU; these results are referred to as T^3-*-*gpu*.

6. Results and Discussion

Overall the results indicate that, as one might expect, there is very little if any significant difference in terms of absolute accuracy between the acoustic models trained with Sphinx versus those trained with HTK. Nevertheless in all the evaluations the RTF versus Word Accuracy metric clearly favors the WFST-based decoders. Below we summarize the results of the individual evaluations.

The results for the *nov92-5k* task are shown in Figure 2. Here the Sphinx3 and HDecode results mirror those reported in [6], and the T^3 results effectively duplicate our own findings in [1]. We also see that Juicer performs very similarly to the T^3-*-*sse* cascades, achieving the same absolute accuracy at a negligibly higher average RTF.

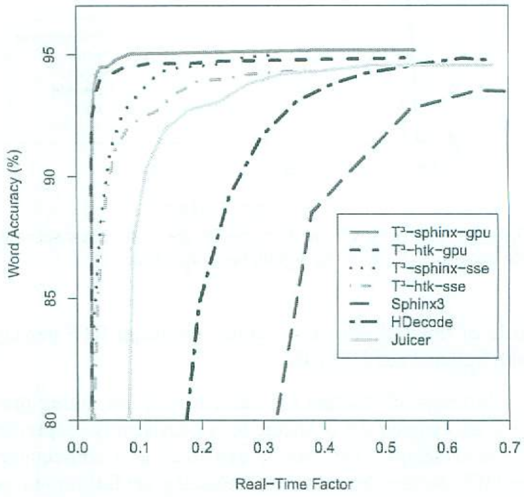

Figure 2: RTF versus Word Accuracy on the nov'92 WSJ test set using the 5k vocabulary bigram language models.

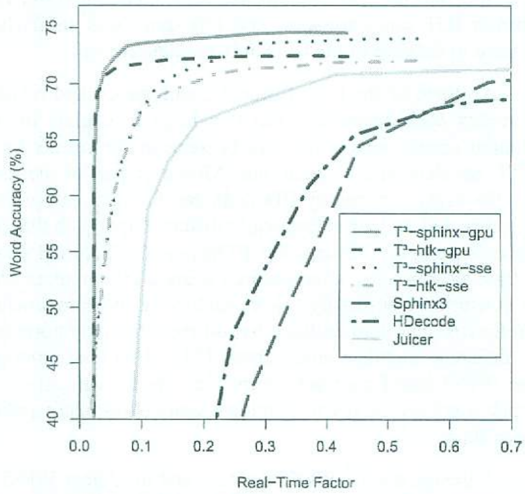

Figure 3: RTF versus Word Accuracy on the si_dt_s2 test set using the 20k vocabulary bigram language models.

On the more difficult *si_dt_s2-20k* task, the results of which are described in Figure 3, we see that the Sphinx acoustic model based cascade wins out slightly in terms of absolute accuracy, but on the whole the results are quite consistent with those of *nov92-5k*. We do note however, that despite eventually achieving the same absolute accuracy, in this case HDecode incurs a significant slow down in terms of average RTF, while Sphinx3 remains relatively unchanged in this regard.

Finally on the *si_dt_s2-64k* task, the results of which are shown in Figure 4, although the WFST decoders remain nearly unchanged in terms of RTF versus Word Accuracy, and the tree-based decoders prove to be equally accurate, we see that the

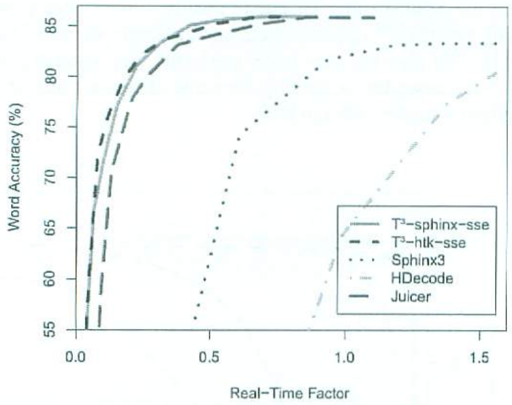

Figure 4: RTF versus Word Accuracy on the si_dt_s2 test set using the 64k vocabulary CSR-based 3-gram language model.

scale of this task imposes a quite significant RTF penalty on both Sphinx3 and HDecode.

In terms of absolute accuracy, it is worth noting that the GPU accelerated T^3 configurations consistently outperformed the other setups. This can be explained as a consequence of the GPU accelerated systems performing the full logsum when computing the mixture scores, as opposed to the somewhat less computationally intensive yet also less accurate logmax operation which is employed for the CPU only configurations. When enabling the GPU, both Sphinx and HTK models exhibited very similar RTF characteristics, and this trend was similarly reflected in the case of the CPU only configurations.

All three of the tasks featured clean speech and relatively complex acoustic models, and this helps to explain the substantial speed gains which can be seen in the results for the GPU accelerated configurations. Moreover part of the reason for the delay incurred by HDecode and Sphinx3 can probably be attributed to the fundamentally different approach that these decoders employ. Essentially HDecode and Sphinx3 employ a time synchronous Viterbi search using a search space which is constructed on-the-fly. Here, each node in a pronunciation prefix tree can hold multiple hypotheses that may correspond to different language model states [14]. This is in contrast to the WFST-based approach where the deterministic, static network topology serves to eliminate ambiguous input paths at each state.

Although the T^3 WFST decoder and the Juicer WFST decoder produce quite similar RTF vs. Word Accuracy results, we note that they exhibit quite different memory consumption characteristics. Specifically, in the case of the *si_dt_s2-64k* task, T^3 required approximately 8.5GB of memory during decoding, while Juicer used an average of 13GB for the same cascade. For T^3 these characteristics were nearly identical across the Sphinx and HTK-based cascades. On the other hand the tree-based decoders utilized an average of about 1.1GB for the same models on this task, making it quite clear that the RTF versus Word Accuracy trade-off is something that must be carefully considered when choosing a decoding framework. We note, for example, that where memory and storage are not an issue, the significant speed advantages of the WFST decoders may often facilitate the application of further post-processing operations without incurring any perceptible change in response from a user perspective.

7. Conclusion and Future Work

In this paper we have presented an empirical comparison of the T^3 decoder alongside the HDecode, Sphinx3 and Juicer decoders, and further shown that it performs quite favorably in terms of RTF and Word Accuracy. We have shown that an important feature of the T^3 decoder is the ability to operate on HTK or Sphinx models with comparable performance, and we have further confirmed the supplementary gains that may be achieved through use of the GPU.

In future work we plan to conduct further evaluations comparing a yet wider range of existing decoders and decoding tasks. If the Sphinx and HTK models generate complementary errors we plan to investigate a tightly coupled combination T^3 system that can operate on both sets of models.

Finally, although every effort was made to ensure that the auxiliary parameter tuning process was fair for each of the systems examined, we acknowledge that due to the manual tuning method employed, there may still be suboptimal values in use for one or more of the decoders. In future we would like to investigate the application of automatic parameter optimization techniques such as those described in [14] as these may provide an even stronger empirical framework for further experimentation.

8. References

[1] Novak, J., Dixon, P., Furui, S., "An Empirical Comparison of Sphinx and HTK models for Speech Recognition", in Proc. ASJ 2010, pp. 73-74, Mar. 2010.

[2] Dixon, P., Caseiro, D., Oonishi, T., Furui, S., "The Titech Large Vocabulary WFST Speech Recognition System," in Proc. ASRU, pp. 13011304, 2007.

[3] Paul, D., B., Baker, J., M., "The Design for the Wall Street Journal-based CSR Corpus," in Proc. ICSLP 92, pp. 357-362, 1992.

[4] Young, S., Everman, G., Kershaw, D., Moore, G., Odell, J., Ollason, D., Valtchev, V., Woodland, P., "The HTK Book," Cambridge University Engineering, 2006.

[5] Moore, D., Dines, J., Magimai Doss, M., Vepa, O., Cheng, O., Hain, T., "Juicer: A Weighted Finite State Transducer Speech Decoder," in Proc. Interspeech, pp. 241-244, 2005.

[6] Vertanen, K., "Baseline WSJ Acoustic Models for HTK and Sphinx: Training Recipes and Recognition Experiments," Cavendish Laboratory, University of Cambridge, 2006.

[7] Huggins-Daines, D., "Why Compare Sphinx and HTK" Online: http://lima-2.speech.cs.cmu.edu/moinmoin/SphinxHTK, accessed Mar. 2010.

[8] Allauzen, C., Mohri, M., Roark, B., "Generalized Algorithms for Constructing Language Models," in Proc. ACL, pp.4047, 2003.

[9] Allauzen, C., Mohri, M., Riley, M., Roark, B., "A Generalized Construction of Integrated Speech Recognition Transducers," in Proc. ICASSP, pp. 761764, 2004.

[10] Cieri, C., Miller, D., Walker, K., "The Fisher Corpus: a Resource for the Next Generations of Speech-to-Text," in Proc. LREC, pp. 69-71, 2004.

[11] Dixon, P., Novak, J., Furui, S., "Recent Evaluations of a WFST-Based Speech Recognition Decoder.", in IEICE Tech. Report, SP2009-78, 2009.

[12] Doddington, G., "CSR Corpus Development," DARPA SLS Workshop, pp. 363-366, 1992.

[13] Garner, P., Dines, J., Hain, T., El Hannani, A., Karafiat, M., Korchagin, D., Lincoln, M., Wan, V. Zhang, L., "Real-Time ASR from Meetings," in Proc. of Interspeech, 2009.

[14] El Hannani, A., Hain, T., "Automatic Optimization of Speech Decoder Parameters," IEEE Signal Processing Letters, 2009.