Comparing Open-Source Speech Recognition Toolkits *

Christian Gaida[1], Patrick Lange[1,2,3], Rico Petrick[2], Patrick Proba[4], Ahmed Malatawy[1,5], and David Suendermann-Oeft[1]

[1] DHBW, Stuttgart, Germany
[2] Linguwerk, Dresden, Germany
[3] Staffordshire University, Stafford, UK
[4] Advantest, Boeblingen, Germany
[5] German University in Cairo, Cairo, Egypt

Abstract. In this paper, a large-scale evaluation of open-source speech recognition toolkits is described. Specifically, HTK in association with the decoders HDecode and Julius, CMU Sphinx with the decoders pocketsphinx and Sphinx-4, and the Kaldi toolkit are compared in terms of usability and expense of recognition accuracy. The evaluation presented in this paper was done on German and English language using respective the Verbmobil 1 and the Wall Street Journal 1 corpus.
We found that, Kaldi providing the most advanced training recipes gives outstanding results out of the box, the Sphinx toolkit also containing recipes enables good results in short time. HTK provides the least support and requires much more time, knowledge and effort to obtain results of the order of the other toolkits.

1 Introduction

Over the last decades, a good number of continuous speech recognition systems have seen the light of day. On the one hand, there are commercial systems such as AT&T Watson [1], Microsoft Speech Server [2], Google Speech API [3] and Nuance Recognizer [4]. On the other hand, proprietary systems offer little control over the recognizer's features, and limited native integrability into other software, leading to a releasing of a great number of open-source automatic speech recognition (ASR) systems.

Due to the growing number of open-source ASR systems, it becomes increasingly more difficult to understand which of them suits the needs of a given target application best. This paper is to report about the results of a study to compare popular open-source large vocabulary ASR toolkits with support by a community and easy access. We evaluate the effort-performance ratio of the selected ASR

* This work was supported by a grant from the Baden-Wuerttemberg Ministry of Science and Arts as part of the research project OASIS. Furthermore, we wholeheartedly appreciate the substantial advice we received by Nick Shmyrev, a principal contributor to the Sphinx project.

toolkits because a perfectly fair comparision on performance only should result in similar error rates differing only due to different decoder implementations.

We selected the following ASR systems for our evaluation:

– Hidden Markov Model Toolkit (HTK[6]) [5] (v3.4.1) associated with large vocabulary decoders HDecode (v3.4.1) and Julius[7] [6] (v4.3),
– pocketSphinx [7] (v0.8) and Sphinx-4 [8] of the Carnegie Mellon University (CMU) Sphinx family[8] [9], and
– Kaldi[9] [10].

We identified further open-source systems and kits:

– RWTH Aachen Automatic Speech Recognition System (RASR) [11],
– Segmental Conditional Random Field Toolkit for Speech Recognition (SCARF) [12],
– Improved ATROS (iATROS) [13],
– SRI International's Decipher [14],
– idiap's Juicer [15] and
– SHoUT speech recognition toolkit[10],

which we did not select for our evaluation, cause they are not as wide spread in the community or as easy available like the chosen ones and therefore restrict the accessibility.

The present research was carried out in the scope of the project OASIS (*O*pen-source *A*utomatic *S*peech recognition *I*n *S*mart devices) [16], sponsored by the Baden-Wuerttemberg Ministry of Science and Art, suggesting one of the evaluation's language to be German. We are also applied the Wall Street Journal 1 (WSJ1) corpus to our comparison. This is to test the generalizability of our conclusions for the English language and read rather than conversational speech. Ultimate goal of the project is to increase spoken language understanding performance in spoken dialog systems [17].

Related to the present work, there is a comparison [18] evaluating state-of-the-art open-source speech recognition systems on standard corpora, but not including Kaldi, which was developed after this work. Another study [19] was done on free speech recognizers, but is, however, limited to corpora of the domain of virtual human dialog.

The present work features three main contributions:

(i) In extension to [18] we were the first to include Kaldi in a comprehensive comparison of open-source speech recognition toolkits.

(ii) Unlike [19], we are using standard corpora which allows for comparing the achieved performance to other systems and techniques published by the community.

[6] Available from `http://htk.eng.cam.ac.uk/`
[7] Available from `http://julius.sourceforge.jp/`
[8] Available from `http://cmusphinx.sourceforge.net/`
[9] Available from `svn://svn.code.sf.net/p/kaldi/code/`
[10] Available from `http://shout-toolkit.sourceforge.net/`

(iii) As opposed to the corpora used in [19] which are rather small in size (the largest corpus, Blackwell, comprises a training data set of about 81k tokens), our study is based on significantly larger corpora with 285k training tokens for Verbmobil 1 (VM1) [20] and approx 1.3M tokens for Wall Street Journal 1 (WSJ1) [21].

This paper is organized as follows: Section 2 gives an overview of the used corpora. Details on how we trained acoustic models are given in Section 3 and on language models in Section 4. In Section 5, our tuning experiments on the development set and the derived setups of the evaluation are described. Section 6 presents and discusses the results of the test run on the test set. In Section 7, we draw conclusions.

2 Data

2.1 Verbmobil 1 corpus

The German Verbmobil project was funded by the German Ministry of Science and Technology (BMBF) and was carried out in the years 1993 through 2000 [22]. The VM database includes dialogic speech in three languages (English, Japanese, German) in the appointment scheduling task. Statistics of the training, development, and test portions of the Verbmobil 1[11] (VM1) corpus are shown in Table 1.

Table 1. Sets of the Verbmobil 1 corpus.

set	utterances	tokens	types	OOV	OOV tokens		audio
				types	n	%	[h]
training	12,590	285,168	6,452	–	–	–	30.5
development	630	15,084	1,537	235	323	2.1	1.6
test	631	14,615	1,342	173	272	1.9	1.5

2.2 Wall Street Journal 1 corpus

The English Wall Street Journal 1[12] (WSJ1) corpus was published 1994 [21] and includes read speech in English of read out loud parts of Wall Street Journal news. Details of the sets used for training, development and test are shown in Table 2.

[11] Orderable from Bavarian Archive for Speech Signals (BAS), `http://www.phonetik.uni-muenchen.de/Bas/BasVM1eng.html`

[12] Orderable from Linguistic Data Consortium (LDC), `https://catalog.ldc.upenn.edu/LDC94S13A`.

Table 2. Sets of the Wall Street Journal 1 corpus.

set	utterances	tokens	types	OOV	OOV tokens		audio
				types	n	%	[h]
training	78,103	1,299,376	18,114	–	–	–	164.0
development	12,665	221,998	12,601	3,313	5,160	2.3	29.0
test (november '93)	4,878	80,617	5,128	757	1,179	1.5	9.8

3 Acoustic modeling

3.1 HDecode and Julius

The acoustic models for the decoders HDecode and Julius are trained using HTK. This toolkit provides a good documentation [5] and examples of a basic training pipeline, further development of advanced techniques is possible, but requires extensive knowledge and very much time compared to the other toolkits. In most cases the development of an own customized toolchain from scratch is needed, which makes the training a difficult and error-prone task for inexperienced people.

Our training methodology follows the descriptions in the HTK Book [5] and [18]. The HTK tool *HCopy* and speech file manipulation software[13] for sphere files is used to convert the audio data from the National Institute of Standards and Technology (NIST) speech files to feature files with Mel Frequency Cepstrum Coefficients with 0th coefficient, delta and acceleration components and cepstral mean normalization (MFCC_0_D_A_Z). The main coding parameters are shown in Table 3.

Table 3. HCopy audio coding parameters.

parameter	value
TARGETKIND	MFCC_0_D_A_Z
USEHAMMING	T
PREEMCOEF	0.97
NUMCHANS	26
CEPLIFTER	22
NUMCEPS	12

Training of the monophones includes silence handling and realignment to enhance the matching between pronunciations and acoustic data. In every step in training, four iterations of the Baum-Welch reestimation are performed.

After training monophones, several recipies for training word-internal and crossword triphones are used. While word-internal triphones are the default setting, crossword triphones are required for HDecode and tested as an option for

[13] Available from `http://www.itl.nist.gov/iad/mig/tools/`

Julius. For state tying and synthesizing unseen phones, decision-tree-based clustering is used. The mixture splitting for training of Hidden Markov Models with gaussian mixtures is done first in steps to the power of two, later in predefined steps.

The training of crossword triphones, the details of tree-based clustering and the methodology of training models with gaussian mixtures has to be discovered and developed by the user. There are more advanced techniques like speaker adaptation and discriminative training for improving the performance available, but the expense of development of such a toolchain increases significantly and the build up is not possible without extensive knowledge, so we did not use these techniques in our comparison.

3.2 Pocketsphinx and Sphinx-4

The acoustic models of pocketsphinx and Sphinx-4 are trained with the CMU sphinxtrain (v1.0.8) toolkit, providing all necessary tools to make use of the above described features. Scripting is not necessary. All parameters can easily be set in a configuration file but a certain level of expertise in speech recognition is necessary because the documentation is sufficient but not extensive.

Besides continuous acoustic models, sphinxtrain is able to train tied (semi-continuous) Gaussian mixture models. The models are trained with Mel Frequency Cepstral vectors. By default setting they are similar to HTK and consist of 13 cepstral, 13 delta, and 13 acceleration coefficients. Further feature extraction parameters are shown in Table 4. Furthermore, we used Linear Discriminant Analysis with Maximum Likelihood Linear Transformation (LDA+MLLT) to reduce the number of feature dimensions.

Table 4. Sphinx feature extraction parameters.

parameter	value
-feat	1s_c_d_dd
-lifter	22
-transform	dct
-lowerf	130
-upperf	6800

In contrast to pocketsphinx, Sphinx-4 is limited to continuous acoustic models.

3.3 Kaldi

The Kaldi toolkit provides several example pipelines for different corpora like e. g. WSJ1.

The capabilities of these pipelines include LDA+MLLT, speaker adaptive training (SAT), maximum likelihood linear regression (MLLR), feature-space

MLLR (fMLLR) and maximum mutual information (MMI, fMMI). Gaussian Mixture Models (GMM) and Subspace GMM (SGMM) are also supported. Further the training of Deep Neural Networks (DNN) on top of GMM models with layer-wise pre-training based on Restricted Boltzmann Machines, per-frame cross-entropy training, and sequence-discriminative training, using lattice framework and optimizing the State Minimum Bayes Risk criterion [23].

The training is of high computational expense, implementation and pipelines are optimized for parallel computing, the training of DNNs supports the usage of GPUs to significantly speed up the processing.

4 Language modeling

4.1 HDecode and Julius

The language models for HDecode and Julius are ARPA n-gram models. In a first step they were trained with a toolchain build up with HTK tools, but the resulting files could not be used for extern decoders. Finally the CMU language model toolkit [24] was used for language model training. The toolchain contains extracting word frequencies and vocabulary from the input text with *text2wfreq* and *wfreq2vocab* followed by generating the ARPA language model file with *text2idngram* and *idngram2lm*. Custom parameters are e. g. the n-gram order, whether open vocabulary is permitted and the discounting methodology of handling unseen n-grams.

The ARPA file is directly used with HDecode, for Julius it is converted into binary format using the tool *mkbingram* to speed up loading.

4.2 pocketsphinx and Sphinx-4

Sphinx uses the CMU language model toolkit toolchain as described in Section 4.1. The ARPA file have been converted into binary format to speed up decoding.

4.3 Kaldi

Due to the language model's internal representation as finite state transducers, Kaldi requires the conversion of ARPA language models as trained by the aforementioned tools into a decoder-specific binary format. For this purpose, a number of utilities are used including the tools *arpa2fst*, *fstcompile*, and multiple Perl scripts.

5 Experiments

5.1 HDecode and Julius

Language model. In a first step we increased the n-gram order up to four finding that three is the optimal choice on both corpora. Observing best results for an order of three we decided to use these for other decoders. In the test runs on VM1 and WSJ1 we used a ARPA trigram language model for HDecode and Julius. In a second step we compared different discounting strategies for handling unseen n-grams, leading to the result Witten-Bell discounting was the best choice for HDecode on both corpora and Julius on WSJ1. On VM1 Good-Touring discounting was chosen for Julius as the best option.

Acoustic model. Due to a significantly larger number of parameters, the focus of performance tuning is on the acoustic model. The tuning process is a quite difficult task with many degrees of freedom and covering them all is nearly impossible. Among others, we investigated

- the influence of using word-internal triphones vs. crossword triphones for Julius. After running a number of experiments on the development set of VM1, we found that crossword triphones are the better choice for the used corpora.
- the configuration of state-based vs. decision-tree-based clustering. In tree-based state tying, different sets of questions are tested. The main questions differentiate sound groups (vowel, fricative, affricate, sonorant, nasale, plosive, liquide) and extended questions (front, center, back). There are also thresholds for outlier removal (RO) and force of tying, the tree branch threshold (TB) available. In the test run, we used models trained with decision-tree-based clustering. Decoding the VM1 test set the model used with HDecode had 2300 tied states (RO 100, TB 1250), the model of Julius 2660 tied states (RO 100, TB 1000). On the WSJ1 corpus the used model for both decoders had 6571 tied states (RO 100, TB 2500). The choice of these parameters was based on the best results we produced in a quite large amount of tuning runs on the development sets building up a set of accuracy values depending on the parameters.
- the number of mixtures and the splitting strategy. Several development runs showed the optimal numbers of gaussians were 32 (VM1) and 64 (WSJ1) for HDecode, respectively 24 and 48 for Julius. In first runs a mixture splitting to the power of two was used. Using smaller steps increased the performance, so predefined steps (1-2-4-6-8-12-16-24-32-48-64) for increasing the number of gaussians were used later.

Decoder. Also the decoders can be operated with a number of different configurations. For HDecode, optimal parameters included a word insertion penalty of 0, a language model weigth of 10, and a pruning threshold of 150. The decoding parameters of the evaluation of Julius are summarized in Table 5. The parameter values are increasing the usage of computational resources, slowing down the decoding process but raising the recognition accuracy.

5.2 Pocketspinx

Language model. As observed for the HTK related decoders, an arpa trigram language model worked best on the VM1 and WSJ1 corpus.

Table 5. Julius decoding parameters.

parameter	value
beam width (-b)	10000
expanded hypotheses (-m)	10000
candidates (-n)	100
stack size (-s)	20000
force output (-fallback1pass)	

Acoustic model. Due to a significantly larger number of parameters, the focus of performance tuning is on the acoustic model. Among others, we investigated during experiments on VM1

– the influence of semi-continuous and continuous models. In a first evaluation cycle using the default parameters for both model types, we found that the continuous acoustic model outperforms the semi-continuous one by 5.5 % with a WER of 36.10 % on the development set.
– the number of mixtures and senones and the reduction of the number of feature vector dimensions by means of LDA+MLLT. Reducing the number of dimensions from 39 to 32 results in a WER improvement of up to 2 % WER as shown in Table 6. In the test run on VM1, a model with 64 mixtures per senone, 1750 senones, and with LDA+MLLT was used.

Table 6. WER of pocketsphinx on the VM1 development set.

GMM per senone	senones	LDA+MLLT	WER
8	200	no	36.1
16	3000	no	25.0
64	1750	no	25.2
64	1750	yes	23.2
64	2000	no	24.9
64	2000	yes	23.7

The parameters for the test run on WSJ1 were derived from [18], a model with 32 mixtures per senone, 8000 senones, and LDA+MLLT was used.

Decoder. In all experiments, we used the decoder's default parameters with the exception of the feature extraction parameters shown in Table 4. Further details on our pocketsphinx tuning setup can be found in [25].

5.3 Sphinx-4

Language model. For VM1 we used the same ARPA trigram model trained for pocketsphinx throughout all experiments, for WSJ1 the 20k trigram model[14] provided by CMU was used.

Acoustic model. In the test run, a model resulting of the pocketSphinx experiments with 64 mixtures per senone and 2000 senones was used for VM1. For WSJ1 a model provided with Sphinx-4 was used.

Decoder. In all experiments and the test run, we used the decoder's default parameters.

5.4 Kaldi

Language model. In view of VM1 the ARPA trigram model used for pocketsphinx and Sphinx-4 was converted to the finite state transducer representation required by Kaldi, regarding to WSJ1 the predefined training script was used including the generation of a 4-gram language model.

Acoustic model. In all experiments and the test run, we used the predefined training scripts for WSJ1, for VM1 these scripts were only adapted in view of the input data.

In the test run a DNN on top of GMM models with LDA+MLLT+SAT+fMLLR were used for both corpora.

Decoder. In all experiments and the test run, we used the predefined decoding scripts including fMLLR and fMMI.

6 Results

The results of the test runs on the test sets are shown in Table 7.

The time spend to set up, prepare, run and optimize the toolkits was most for HTK, less Sphinx and least Kaldi.

Compared to the other recognizers, the outstanding performance of Kaldi can be seen as a revolution in open-source speech recognition technology. The system features next to all state-of-the-art techniques discussed in literature including LDA, MLLT, SAT, fMLLR, fMMI, and DNNs out of the box. Even being no expert, the provided recipes and scripts enable the usage of all these techniques to the user in short time.

[14] `http://sourceforge.net/projects/cmusphinx/files/Acoustic and Language Models/US English WSJ5K Language Model/WSJ20K_trigram_lm.zip`.

The toolkit of the Sphinx family also come up with a training tool, not containing all techniques of Kaldi leading to less accuracy, but also enabling training and doing speech recognition shortly after installing the system.

HTK is the most difficult toolkit. Setting up the system required the development of the training pipeline, which was time consuming and error-prone. The development of techniques especially them beyond the provided tutorials requires much more knowledge and effort as needed for setting up the other systems. Training techniques like adaptation and discriminative training are basically possible, but the development of the toolchain is nearly impossible without being an expert. The results obtained are similar to Sphinx, but the effort to get these is quite larger.

Table 7. Word error rates on the VM1 test set and the WSJ1 november '93 test set.

recognizer	VM1	WSJ1
HDecode v3.4.1	22.9	19.8
Julius v4.3	27.2	23.1
pocketsphinx v0.8	23.9	21.4
Sphinx-4	26.9	22.7
Kaldi	12.7	6.5

7 Conclusions

In this paper, we described a large scale evaluation of open-source speech recognition toolkits. The main contributions are the inclusion of Kaldi to a comprehensive evaluation of open-source ASR systems and the usage of standard corpora making our results compareable to other publications in the field.

We trained language and acoustic models for HDecode, Julius, pocketsphinx, Sphinx-4, and Kaldi. We partially tuned the recognition systems and ran tests on the German Verbmobil 1 and the English Wall Street Journal 1 corpus.

Our experiments show an order of the evaluated toolkits regarding to the ratio of effort to performance. Kaldi outperforms all the other recognition toolkits, providing training and decoding pipelines including the most advanced techniques out of the box. This conveniently enables the best results in short time but has the highest computational cost. The Sphinx toolkit also provides a training pipeline, not as advanced as the Kaldi pipeline, but with the possibility to generate good results shortly. HTK comes with the simplest start-up kit, the buildup of a self-trained recognition system to reach the performance of e. g. Sphinx requires the development of advanced parts by the user and extensive tuning.

So after installing and compiling the toolkits when the users of Kaldi and Sphinx obtain there first results the HTK users are still scripting their training pipelines.

References

1. Goffin, V., Allauzen, C., Bocchieri, E., Hakkani-Tür, D., Ljolje, A., Parthasarathy, S., Rahim, M., Riccardi, G., Saraclar, M.: The AT&T WATSON speech recognizer. In: Proc. of the ICASSP, Philadelphia, USA (2005)
2. Dunn, M.: Pro Microsoft Speech Server 2007: Developing Speech Enabled Applications with .NET (Pro). Apress, Berkeley, USA (2007)
3. Adorf, J.: Web Speech API. Technical report, KTH Royal Institute of Technology, Stockholm, Sweden (2013)
4. Nuance Communication Inc.: Speech recognition solutions. www.nuance.com/for-business/by-solution/speech-recognition/ (2014)
5. Young, S., Evermann, G., Gales, M., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., Woodland, P.: The HTK Book, Version 3.4. Cambridge University, Cambridge, UK (2006)
6. Lee, A., Kawahara, T.: Recent development of open-source speech recognition engine Julius. In: Proc. of the APSIPA ASC, Los Angeles, USA (2009)
7. Huggins-Daines, D., Kumar, M., Chan, A., Black, A., Ravishankar, M., Rudnicky, A.: Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices. In: Proc. of the ICASSP, Toulouse, France (2006)
8. Lamere, P., Kwok, P., Gouvea, E., Raj, B., Singh, R., Walker, W., Warmuth, M., Wolf, P.: The CMU SPHINX-4 Speech Recognition System. In: Proc. of the ICASSP'03, Hong Kong, China (2003)
9. Lee, K.F., Reddy, R.: Automatic Speech Recognition: The Development of the Sphinx Recognition System. Kluwer Academic Publishers, Norwell, MA, USA (1988)
10. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., Vesely, K.: The Kaldi speech recognition toolkit. In: Proc. of the ASRU, Hawaii, USA (2011)
11. Rybach, D., Hahn, S., Lehnen, P., Nolden, D., Sundermeyer, M., Tüske, Z., Wiesler, S., Schlüter, R., Ney, H.: RASR—The RWTH Aachen University open source speech recognition toolkit. In: Proc. of the ASRU, Hawaii, USA (2011)
12. Zweig, G., Nguyen, P.: SCARF: A segmental conditional random field toolkit for speech recognition. In: In Proc. of the Interspeech, Makuhari, Japan (2010)
13. Lujn-Mares, M., Tamarit, V., Alabau, V., Martnez-Hinarejos, C.D., i Gadea, M.P., Sanchis, A., Toselli, A.H.: iatros: A speech and handwritting recognition system. In: V Jornadas en Tecnologas del Habla (VJTH'2008). (2008) 75–78
14. Murveit, H., Cohen, M., Price, P., Baldwin, G., Weintraub, M., Bernstein, J.: SRI's DECIPHER system. In: Proc. of the Workshop on Speech and Natural Language. HLT '89, Stroudsburg, PA, USA, Association for Computational Linguistics (1989) 238–242
15. Moore, D., Dines, J., Magimai-Doss, M., Vepa, J., Cheng, O., Hain, T.: Juicer: A weighted finite-state transducer speech decoder. In: 3rd Joint Workshop on Multimodal Interaction and Related Machine LEarning Algorithms MLMI'06. (may 2006) IDIAP-RR 06-21.
16. Baden-Wuerttemberg Cooperative State University Stuttgart, Germany: OASIS—Open-Source Automatic Speech Recognition In Smart Devices. (2014) http://www.dhbw-stuttgart.de/themen/kooperative-forschung/fakultaet-technik/oasis.html.

17. Suendermann, D., Liscombe, J., Dayanidhi, K., Pieraccini, R.: A Handsome Set of Metrics to Measure Utterance Classification Performance in Spoken Dialog Systems. In: Proc. of the SIGdial, London, UK (2009)
18. Vertanen, K.: Baseline WSJ Acoustic Models for HTK and Sphinx: Training Recipes and Recognition Experiments. Technical report, University of Cambridge, Cambridge, UK (2006)
19. Yao, X., Bhutada, P., Georgila, K., Sagae, K., Artstein, R., Traum, D.: Practical evaluation of speech recognizers for virtual human dialogue systems. In: Proc. of the LREC, Malta (2010)
20. Florian Schiel: Verbmobil I - VM1. Bavarian Archive for Speech Signals, Munich, Germany. (2012) http://www.phonetik.uni-muenchen.de/Bas/BasVM1eng.html.
21. Linguistic Data Consortium Philadelphia, USA: CSR-II (WSJ1) Complete. (1994) http://catalog.ldc.upenn.edu/LDC94S13A.
22. Wahlster, W., ed.: Verbmobil: Foundations of Speech-to-Speech Translation. Springer, Berlin, Heidelberg, Germany (2000)
23. Rath, P.S., Povey, D., Vesely, K., Cernocky, J.: Improved feature processing for deep neural networks. In: Proc. of Interspeech 2013, International Speech Communication Association (2013) 109–113
24. Clarkson, P., Rosenfeld, R.: Statistical language modeling using the CMU-Cambridge toolkit. In: Proc. of the Eurospeech, Rhodes, Greece (1997)
25. Lange, P., Suendermann-Oeft, D.: Tuning Sphinx to outperform Google's speech API. In: Proc. of the ESSV. (2014)