INTERSPEECH 2009 BRIGHTON

The RWTH Aachen University Open Source Speech Recognition System

David Rybach, Christian Gollan, Georg Heigold, Björn Hoffmeister, Jonas Lööf, Ralf Schlüter, Hermann Ney

Human Language Technology and Pattern Recognition
Computer Science Department, RWTH Aachen University, Germany,
rwthasr@i6.informatik.rwth-aachen.de

Abstract

We announce the public availability of the RWTH Aachen University speech recognition toolkit. The toolkit includes state of the art speech recognition technology for acoustic model training and decoding. Speaker adaptation, speaker adaptive training, unsupervised training, a finite state automata library, and an efficient tree search decoder are notable components. Comprehensive documentation, example setups for training and recognition, and a tutorial are provided to support newcomers.
Index Terms: speech recognition, LVCSR, software

1. Introduction

The interest in speech recognition technology has grown over the last years. For researchers it requires a lot of effort to develop a speech recognition system from scratch, which impedes innovations. Publicly available toolkits, often published under an open source license, facilitate the introduction to research in this area. A couple of open source system are available, for example the HTK Toolkit [1], Sphinx-4 [2], and Julius [3].

Our speech recognition system has been designed for the special requirements of research applications. On the one hand it should be very flexible, to allow for rapid integration of new methods, and on the other hand it has to be efficient, so that new methods can be studied on real-life tasks in reasonable time and system tuning is feasible. The flexibility is achieved by a modular design, where most components are decoupled from each other and can be replaced at runtime. The API is subdivided into several modules and allows for an integration of (high and low level) methods in external applications.

The applicability of our toolkit to real-life tasks has been proven by building several large vocabulary systems in recent international research projects. The European English and Spanish recognition systems developed during the TC-STAR Project are based on our RWTH ASR toolkit [4]. These two systems achieved the best results in the 2007 TC-STAR Evaluation Campaign. Other challenging tasks, like building competitive recognition systems in the GALE project with very large vocabulary for Arabic [5] or acoustic model training with thousands of hours of speech data for a Chinese recognizer [6], have been fulfilled successfully.

A good example for the flexibility of our toolkit is the expeditious development of systems for continuous sign language recognition using video input [7] and for handwriting recognition [8]. Only the feature extraction had to be replaced to adapt the system to these tasks.

An important aspect for developing a system for a large vocabulary task is the support for grid-computing. Nearly all processing steps for acoustic model training and decoding can be distributed in a cluster computer environment. The parallelization scales very well, because we divide the computations on the segment level, which requires synchronization only at the end of the computation.

The toolkit is available for download on our website[1]. We publish our toolkit under an open source license, called "RWTH ASR License", which is derived from the Q Public License v1.0 This license grants free usage including re-distribution and modification for non-commercial use. Publications of results obtained through the use of original or modified versions of the software have to cite this paper.

In the remainder of this paper we describe the individual parts of the framework. First we depict the acoustic front-end and the used models. Then we present the decoder and the finite-state automata library and finally the documentation and supplementary materials.

2. Signal Analysis

Methods for signal analysis are implemented in a generic framework, called Flow, which is described in the next section. The predefined acoustic features computed using this framework are defined in the following section.

2.1. Flow Networks

The Flow module offers a generic framework for data processing. The data flow is modeled by links connecting several nodes to a network. Each node performs some type of data manipulation including loading, storing, and caching of data.

The networks are created at runtime based on a network definition in XML documents, which makes it possible to implement or modify data processing tasks without modifying and re-compiling the software.

Flow networks are used to compute acoustic features as well as to generate and process dynamic time alignments, i.e. mappings from acoustic features to HMM states. Using a caching mechanism, which is also implemented as a node, acoustic features and alignments can be re-used in processing steps requiring multiple iterations.

2.2. Acoustic Features

The basic nodes in a Flow network implement the reading of waveforms from audio files, computing an FFT, miscellaneous vector operations, and different types of signal normalization. The networks included in the toolkit compute MFCC features and a voicing feature. The voicing feature expresses the voicedness of a time frame by detecting the quasi periodic oscillation of the vocal chords [9].

Temporal context can be incorporated by using derivatives of the acoustic features. Another method, commonly used in our systems, is to use a linear discriminant analysis for this purpose. Therefore, consecutive feature vectors in a sliding window are concatenated and projected to a lower dimension [10].

The flexibility of the Flow module allows for an easy implementation of other acoustic features and for the integration of externally computed features.

[1] www-i6.informatik.rwth-aachen.de/rwth-asr

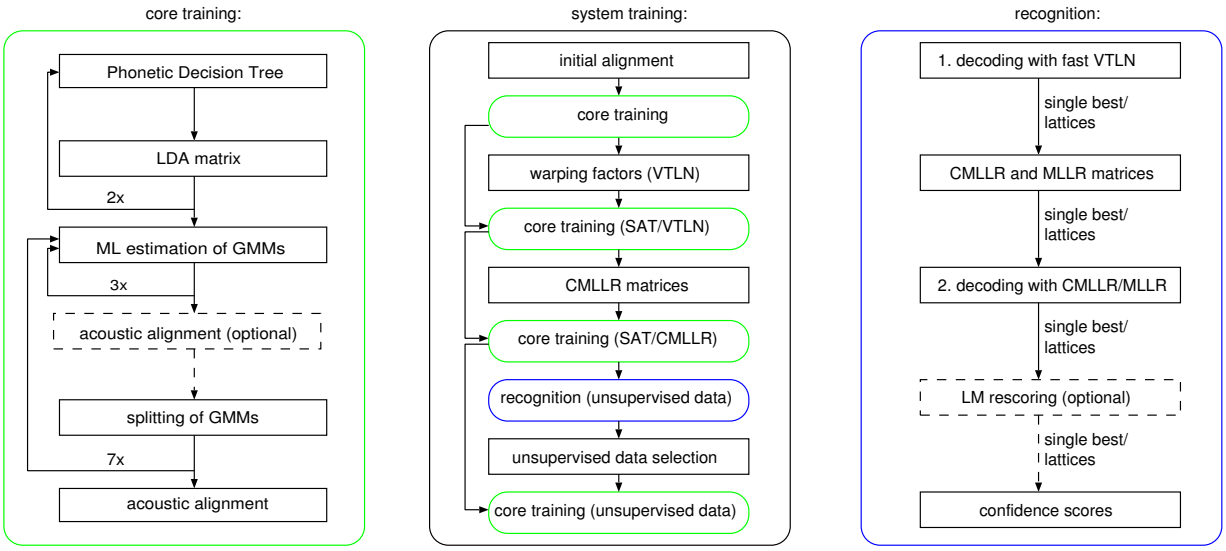

Figure 1: *Example setups for acoustic model training (from scratch) and recognition.*

3. Acoustic Modeling

The acoustic model consists of the transition, the emission, and the pronunciation model. The pronunciation model gives for each word in the vocabulary a list of pronunciations together with a probability of the occurrence. A pronunciation is modelled by a sequence of context dependent phonemes. In the current version, the context is limited to triphones, including context across words.

The toolkit supports strict left-to-right HMM topologies, each representing a context dependent phoneme. Except for silence, which is modeled by a single state, all HMMs consist of the same number of states. The transition model implements loop, forward, and skip transitions. The existing toolkit supports a global transition model which distinguishes only the silence state. Transitions leaving a word are penalized with an extra cost, the word penalty.

The emission probability of an HMM state is represented by a Gaussian mixture model (GMM). By default, globally pooled variances are used. However, several other tying schemes, including density-specific diagonal covariance matrices are supported. Figure 1 provides an overview on the estimation of GMMs and the interaction with the training of the other acoustic components like LDA or CART.

3.1. State Tying: Phonetic Decision Trees

The toolkit supports three methods for state tying. The monophone tying assigns all triphones which share the same central phoneme to the same Gaussian mixture. This is usually used for initializing the acoustic model training. The look-up table mode allows to define a state tying using external tools which is stored in a simple text format. Alternatively, the built-in classification and regression tree (CART) training can be used to estimate a phonetic decision tree [11].

The configuration of the CART training is flexible and supports a variety of phonetic decision tree based tyings. For example, English systems usually perform best when estimating a separate tree for each combination of central phoneme and HMM state. On the other hand, languages like Mandarin benefit from applying a less strict separation. Both and more configurations are supported by the CART estimation tool.

3.2. Confidence Scores

In the speech decoding process different dynamic pruning methods are applied to restrict the set of competing word sequences. This set can be efficiently represented by a lattice (c.f. Section 5). Using the forward-backward algorithm, we efficiently compute the relation between the competing hypotheses by estimating the lattice link posterior probabilities [12]. Depending on the lattice link labels and the structure of the lattice it is possible to compute confidence scores for different units, e.g. word, pronunciation, or HMM state confidence scores.

For the unsupervised refinement or re-estimation of the acoustic model parameters (unsupervised training) the toolkit supports the generation and processing of confidence weighted state alignments. Confidence thresholding on state level is supported for unsupervised training as well as for unsupervised adaptation methods. The toolkit supports different types of state confidence scores, all described in [13]. The emission model can be re-estimated based on the automatically annotated observations and their assigned confidence weights, as presented in [14].

3.3. Speaker Normalization and Adaptation

For state of the art performance in large vocabulary continuous speech recognition, speaker normalization and adaptation are required. The toolkit supports three different methods: Vocal tract length normalization (VTLN) [15, 16], maximum likelihood linear regression (MLLR) [17], and feature space MLLR (fMLLR) (also known as constrained MLLR, CMLLR) [18].

VTLN is a speaker normalization technique aimed at compensating for systematic differences in formant position between different speakers, due to different vocal tract length. For the toolkit, a VTLN is implemented as a parametric linear warping of the MFCC filter bank, as described in [15]. The parameter is estimated using maximum likelihood. Support for one pass, or so called *fast* VTLN [16], is also included, in the form of support for choosing the warping factor using a Gaussian mixture model classifier.

The fMLLR consists of normalizing the acoustic features by the use of a maximum likelihood estimated affine transform, as described in [18]. As an extension, the estimation of dimension reducing affine transforms, as described in [19], is supported.

Both VTLN and fMLLR are implemented in the feature extraction front-end, allowing for use in both recognition and in training, thus supporting speaker adaptive training.

Finally, Maximum Likelihood Linear Regression (MLLR) [17] is supported, which is applying affine transforms to the means of the acoustic model. A regression class tree approach [20] is used to adjust the number of regression classes to the amount of adaptation data available. As a variation, it is possible to do adaptation using only the offset part (and not the matrix part) of the affine transform.

All the adaptation methods can be utilized both for unsupervised and supervised adaptation. Both fMLLR and MLLR estimation can make use of weighted observations, as produced by the confidence measures described in the previous section, allowing for confidence based unsupervised adaptation.

4. Language Modeling

The toolkit does not include tools for the estimation of language models. However, the decoder supports N-gram language models in the ARPA format, produced e.g. by the SRI Language Modeling Toolkit [21]. The order of the language model is not limited by the decoder. Class language models, defined on word classes instead of words, are supported as well. Alternatively, a weighted finite state automaton representing a (weighted) grammar can be used.

5. Decoder

The decoder included in our toolkit is based on the word conditioned tree search [22]. Word conditioned tree search is a one-pass dynamic programming algorithm which uses a precompiled lexical prefix tree as representation of the pronunciation dictionary. When using a tree lexicon, the word identity is not known until a leaf node is reached. Therefore, the language model (LM) probability can only be applied at the word end, although an early incorporation of the LM can be achieved using LM look-ahead. To make the application of the dynamic programming principle possible, the search space has to be structured by introducing separate copies of the lexical tree for each preceding word sequence. The length of this word sequence depends on the order of the language model used, e.g. for a bigram language model only the direct predecessor word is required.

The search space would be too large to be constructed as a whole, instead only the active portions are constructed dynamically in combination with a beam search. The beam search strategy retains for every time step only the most promising hypotheses. Hypotheses with a too low score compared to the best state hypothesis are eliminated by *acoustic pruning*. The beam width, i.e. the number of surviving hypotheses, is defined by a threshold. *Language model pruning* is applied to the word start hypotheses after applying the language model, which limits the number of active tree copies. In addition, histogram pruning restricts the absolute number of active hypotheses.

The acoustic pruning can be refined by incorporating the language model probabilities as early as possible using a language model look-ahead [23]. The anticipated language model probability for a certain state in the tree is approximated by the best word end reachable. This probability is incorporated in the pruning process by combining it with the probability of the state hypothesis.

The tree lexicon is constructed using the HMM-state sequences of the pronunciations of the words in the vocabulary. By using the tied states of the (context dependent) phoneme models, the tree size can be further reduced. For across-word context dependent phoneme models the word beginnings and ends have to be expanded to allow for all possible successor phonemes [24].

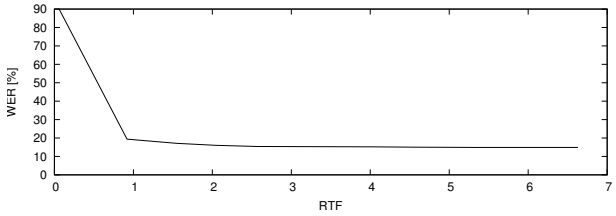

Figure 2: *Recognition quality (word error rate, WER) and runtime (real-time factor, RTF) for the EPPS English task*

The decoder can also generate a word graph (also called lattice) which is a compact representation of the set of alternative word sequences with corresponding word boundaries [25]. This word graph can be used in later processing steps. Our system produces word graphs as finite state automata with attached word boundaries or alternatively in the HTK standard lattice format.

The computation of acoustic likelihoods can be optionally accelerated by the use of SIMD instructions provided by modern processors [26]. The acoustic feature vectors as well as the means of the Gaussian mixture models are transformed to integers using a scalar quantization. The following computations on these quantized vectors are performed using MMX or SSE2 instructions.

5.1. Experimental Results

Using a baseline 1-pass 60k-words recognition system for the automatic transcription of European Parliament Plenary Sessions (EPPS) in English as described in [4], a recognition quality of about 15% word error rate is achievable with a real-time factor of $\sim$4 (see Figure 2) on the evaluation corpus of the 2007 TC-STAR Evaluation Campaign. This corpus consists of 2.9h of speech with an out-of-vocabulary rate of 1.1%. Using further optimization steps, including speaker adaptive training, MPE-training (not yet included), and speaker adaptation, the word error rate can be decreased to 10.6%.

6. Finite-state Automata

Finite-state automata are used in various applications in the field of natural language processing, including speech recognition. In the RWTH ASR toolkit finite-state automata are used for several tasks. The computation of dynamic time alignments, required for acoustic model training and speaker adaptation, uses automata for the construction and representation of the search space. Furthermore, the word lattices generated by the speech recognizer are represented by finite-state automata. Therefore, the lattices generated can easily be post-processed by algorithms defined on weighted finite state transducers.

Finite-state automata are handled by the included RWTH FSA Toolkit [27], which is also available separately under an open source license. The FSA toolkit consists of efficient implementations of data structures and a wide range of well known algorithms for creating and manipulating weighted finite-state automata. On-demand computations can be used for increased memory efficiency.

7. Documentation

The documentation of RWTH ASR is divided into two parts: usage documentation and source code documentation. While the source code documentation is helpful for extending the software, the usage documentation is more comprehensive and more relevant for the normal user.

The usage documentation is organized in a wiki and covers all steps of the acoustic model training, multi-pass recognition, and describes the common concepts of the software and the used

file formats. Emerging questions can be asked in a support forum.

For a quick introduction, we created a step-by-step recipe for the development of a small (100 words) recognizer based on the CMU Census Database (freely available) by the Carnegie Mellon University. The acoustic model training includes estimation of mixture models for triphones, LDA, and CART. All needed configuration files and scripts are publicly available and can be used for research purposes.

To demonstrate a large vocabulary system we offer the acoustic model (triphones, 900K densities), the 4-gram language model (7.5M multi-grams), and the pronunciation dictionary (60K words) developed for our EPPS English system together with a ready-to-use one-pass recognition setup.

8. Conclusion

We announced the release of a new open source speech recognition toolkit. Using this toolkit it is possible to develop state of the art speech recognizers. The various methods and technologies implemented can be used, exchanged, and expanded in a very flexible way. A comprehensive documentation facilitates the usage for novices and users of other systems.

9. Acknowledgements

We thank Maximilian Bisani, Stephan Kanthak, and András Zolnay, who designed and implemented the main parts of this toolkit, as well as all other (former) members of the RWTH ASR group, who contributed code and knowledge.

10. References

[1] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book*. Cambridge University Engineering Department, 2006.

[2] W. Walker, P. Lamere, P. Kwok, R. S. Bhiksha Raj, E. Gouvea, P. Wolf, and J. Woelfel, "Sphinx-4: A flexible open source framework for speech recognition," Sun Microsystems, Inc, Tech. Rep. SMLI TR-2004-139, Nov. 2004.

[3] A. Lee, T. Kawahara, and K. Shikano, "Julius – an open source real-time large vocabulary recognition engine," in *Proc. European Conf. on Speech Communication and Technology*, Aalborg, Denmark, Sep. 2001, pp. 1691–1694.

[4] J. Lööf, C. Gollan, S. Hahn, G. Heigold, B. Hoffmeister, C. Plahl, D. Rybach, R. Schlüter, and H. Ney, "The RWTH 2007 TC-STAR evaluation system for European English and Spanish," in *Proc. Int. Conf. on Speech Communication and Technology*, Antwerp, Belgium, Aug. 2007, pp. 2145–2148.

[5] D. Rybach, S. Hahn, C. Gollan, R. Schlüter, and H. Ney, "Advances in Arabic broadcast news transcription at RWTH," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, Kyoto, Japan, Dec. 2007, pp. 449–454.

[6] C. Plahl, B. Hoffmeister, M.-Y. Hwang, D. Lu, G. Heigold, J. Lööf, R. Schlüter, and H. Ney, "Recent improvements of the RWTH GALE Mandarin LVCSR system," in *Proc. Int. Conf. on Speech Communication and Technology*, Brisbane, Australia, Sep. 2008, pp. 2426–2429.

[7] P. Dreuw, D. Rybach, T. Deselaers, M. Zahedi, and H. Ney, "Speech recognition techniques for a sign language recognition system," in *Proc. Int. Conf. on Speech Communication and Technology*, Antwerp, Belgium, Aug. 2007, pp. 2513–2516.

[8] P. Dreuw, D. Rybach, C. Gollan, and H. Ney, "Writer adaptive training and writing variant model refinement for offline Arabic handwriting recognition," in *Proc. Int. Conf. on Document Analysis and Recognition*, Barcelona, Spain, Jul. 2009.

[9] A. Zolnay, R. Schlüter, and H. Ney, "Extraction methods of voicing feature for robust speech recognition," in *European Conference on Speech Communication and Technology*, vol. 1, Geneva, Switzerland, Sep. 2003, pp. 497–500.

[10] R. Haeb-Umbach and H. Ney, "Linear discriminant analysis for improved large vocabulary continuous speech recognition," San Francisco, CA, USA, p. 1316, Mar. 1992.

[11] K. Beulen, E. Bransch, and H. Ney, "State-tying for context dependent phoneme models," in *Proc. European Conf. on Speech Communication and Technology*, vol. 3, Rhodes, Greece, Sep. 1997, pp. 1179–1182.

[12] G. Evermann and P. Woodland, "Large vocabulary decoding and confidence estimation using word posterior probabilities," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, Jun. 2000, pp. 1655 – 1658.

[13] C. Gollan and M. Bacchiani, "Confidence scores for acoustic model adaptation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Las Vegas, NV, USA, Apr. 2008, pp. 4289–4292.

[14] C. Gollan and H. Ney, "Towards automatic learning in LVCSR: Rapid development of a Persian broadcast transcription system," in *Interspeech*, Brisbane, Australia, Sep. 2008, pp. 1441–1444.

[15] E. Eide and H. Gish, "A parametric approach to vocal tract length normalization," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 1, Atlanta, GA, USA, May 1996, pp. 346 – 349.

[16] L. Welling, S. Kanthak, and H. Ney, "Improved methods for vocal tract normalization," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 2, Phoenix, AZ, USA, Mar. 1999, pp. 761 – 764.

[17] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer Speech and Language*, vol. 9, no. 2, pp. 171 – 185, Apr. 1995.

[18] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech and Language*, vol. 12, no. 2, pp. 75 – 98, Apr. 1998.

[19] J. Lööf, R. Schlüter, and H. Ney, "Efficient estimation of speaker-specific projecting feature transforms," in *Proc. Int. Conf. on Spoken Language Processing*, Antwerp, Belgium, Aug. 2007, pp. 1557 – 1560.

[20] C. Leggetter and P. Woodland, "Flexible speaker adaptation using maximum likelihood linear regression," in *Proc. ARPA Spoken Language Technology Workshop*, Austin, TX, USA, Jan. 1995, pp. 104 – 109.

[21] A. Stolcke, "SRILM - an extensible language modeling toolkit," in *Proc. Int. Conf. on Spoken Language Processing*, Denver, CA, USA, Sep. 2002.

[22] H. Ney and S. Ortmanns, "Progress in dynamic programming search for LVCSR," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1224–1240, Aug. 2000.

[23] S. Ortmanns and H. Ney, "Look-ahead techniques for fast beam search," *Computer Speech and Language*, vol. 14, no. 1, pp. 15–32, Jan. 2000.

[24] A. Sixtus and H. Ney, "From within-word model search to across-word model search in large vocabulary continuous speech recognition," *Computer Speech and Language*, vol. 16, no. 2, pp. 245–271, May 2002.

[25] S. Ortmanns, H. Ney, and X. Aubert, "A word graph algorithm for large vocabulary continuous speech recognition," *Computer Speech and Language*, vol. 11, no. 1, pp. 43–72, Jan. 1997.

[26] S. Kanthak, K. Schütz, and H. Ney, "Using SIMD instructions for fast likelihood calculation in LVCSR," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, Jun. 2000, pp. 1531–1534.

[27] S. Kanthak and H. Ney, "FSA: An efficient and flexible C++ toolkit for finite state automata using on-demand computation," in *Proc. Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, Jul. 2004, pp. 510–517.