BioKIT - Real-time decoder for biosignal processing

Dominic Telaar, Michael Wand, Dirk Gehrig, Felix Putze,
Christoph Amma, Dominic Heger, Ngoc Thang Vu, Mark Erhardt
Tim Schlippe, Matthias Janke, Christian Herff, Tanja Schultz

Cognitive Systems Lab, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
dominic.telaar@kit.edu

10.21437/Interspeech.2014-567

Abstract

We introduce BioKIT, a new Hidden Markov Model based toolkit to preprocess, model and interpret biosignals such as speech, motion, muscle and brain activities. The focus of this toolkit is to enable researchers from various communities to pursue their experiments and integrate real-time biosignal interpretation into their applications. BioKIT boosts a flexible two-layer structure with a modular C++ core that interfaces with a Python scripting layer, to facilitate development of new applications. BioKIT employs sequence-level parallelization and memory sharing across threads. Additionally, a fully integrated error blaming component facilitates in-depth analysis. A generic terminology keeps the barrier to entry for researchers from multiple fields to a minimum. We describe our online-capable dynamic decoder and report on initial experiments on three different tasks. The presented speech recognition experiments employ Kaldi [1] trained deep neural networks with the results set in relation to the real time factor needed to obtain them.

Index Terms: biosignal processing, toolkit, dynamic decoder, speech recognition

1. Introduction

In many fields of research, relevant information in sensor data can be modeled by sequences of atomic states using the statistical principles of Hidden Markov Models (HMMs). Decoding an input signal stream, i.e. finding such sequences in time series, is a key pattern recognition approach that is relevant for numerous applications.

Structural information, as used in automatic speech recognition (ASR) in form of statistical language models, can add information and help to restrict the search space for efficient decoding. Over the last 35 years, a massive effort of the automatic speech processing community has pioneered HMM-based modeling and decoding, which led to effective and efficient algorithms. Yet, this technology is not easily accessible for researchers from other disciplines, as the available tools include specific optimizations and require researchers to transfer their concepts to those of the toolkit's original target community. The lack of a toolkit that met all of our requirements led to the development of BioKIT, which is designed, implemented, and used by our interdisciplinary team of researchers from the fields of speech, electrophysiology, and motion recognition. It has been developed to fulfill the demands of different research areas by complying with the following key requirements.

- **Flexible processing and modeling** of various types of signals, such as multi-channel bio-physiological signals, high-resolution frequency based speech signals, and low-resolution time-based signals like accelerometer signals: All processing modules should have generic interfaces that allow flexible use within arbitrary processing chains for state-of-the-art decoding. Model assumptions should be minimal, e.g. flexible HMM architectures should be supported, including multiple streams, and hierarchical modeling.
- **Fast setup of experiments and flexible code-base:** The efforts of creating a new recognition setup and performing experiments should be small. Code should be designed in a modular fashion making it easy to combine and replace features or change behavior of existing algorithms. This should be possible during run-time and, in particular, without the necessity of recompiling and linking the complete code-base.
- **Processing of big amounts of data:** Sizes of data sets for ASR and other human-machine interaction applications, such as wearable computing and activity recognition, are increasing. Therefore, the toolkit should be able to handle massive amounts of data with correspondingly big models. Furthermore, parallelization which means utilizing clusters and multi-core machines efficiently (i.e. share memory across threads) is a requirement.
- **Online-capability:** The decoder needs to process a continuous data stream in real-time by chunk-wise decoding and outputting of intermediate results.
- **Error analysis:** During research it is important to analyze recognition errors to identify the responsible system components [2, 3] and room for improvements.
- **Accessibility:** The terminology used in the toolkit should be sufficiently generic to appeal to a wide audience.

1.1. Related work

There are a number of toolkits which focus mostly on ASR, e.g. the HTK toolkit [4], Sphinx-4 [5], Julius [6], and more recently the Kaldi toolkit [1]. A toolkit with a broader focus is the RWTH Aachen toolkit [7]. The Georgia Tech Gesture Toolkit [8] extends HTK to simplify the process of building HMM-based gesture recognizers. Notable frameworks which also had an impact on the development of BioKIT are the Janus Recognition Toolkit [9, 10] and the Attila toolkit [11].

Current toolkits usually offer either dynamic [7, 11] or static decoders [1]. When focusing on acoustic modeling research, other components are kept constant and the resulting static network can be reused between experiments [1]. Since all knowledge sources are integrated into the network, the decoder is more streamlined than a dynamic decoder. A drawback is that

only language models which can be converted into a finite state transducer can be used. Additionally, models cannot exceed a certain size to keep network size in check. Dynamic decoders offer more flexibility for language modeling at the expense of more complicated decoders [11]. BioKIT, like Sphinx-4 [5], is a dynamic decoder implementing a token passing algorithm which is described further in Section 3.4. We use a two-layer design [11] and additionally allow users to integrate their own components (such as preprocessing modules) on the fly. This is particularly useful for working with EEG/EMG signals (see Section 2). Building on the work of Chase and Nanjo et al. [2, 3], we directly integrated an error blaming component into BioKIT providing analysis at the state level. Other than the above toolkits, BioKIT allows to order the HMMs in hierarchies and perform a decoding on this hierarchical structure (refer to Section 2.2).

1.2. Terminology

Our terminology abstracts from the specific terminology to appeal to a wide audience in biosignal research. The smallest meaningful unit in our system is an *Atom*, representing a single HMM. A *Token* is a sequence of one or more *Atoms*, which also compose BioKIT's final recognition results. In the simplest case, one *Token* consists of one *Atom* which equates to modeling a *Token* with one distinct HMM. The *Dictionary* describes the mapping of *Tokens* to *Atom* sequences. The *TokenSequenceModel* models occurrence probabilities of *Token* sequences. Emission probabilities of HMM states are computed by the *FeatureVectorScorer*.

To relate these terms to the common ASR terminology: A single *Atom* corresponds to a phone in speech. A sequence of *Atoms* equates to the pronunciation of a *Token*, i.e. a word. The *TokenSequenceModel* conforms to the language model. The *FeatureVectorScorer* corresponds to the acoustic model. For an activity recognition system, *Atoms* could relate to primitive motions such as raising an arm or grabbing an object, whereas *Tokens* would model meaningful sequences of primitive motions such as opening a door. The *TokenSequenceModel* would model the probabilities of observing certain actions or activities in a sequence.

2. Biosignals

The following section lists some current research activities for different biosignals using BioKIT. Each section briefly describes a signal's requirements for a decoder and how we meet them. The term biosignal refers here to signals which are emitted by the human body to control, regulate and transfer information within the human organism or to interact with its environment. Biosignals are measured in different sizes depending on the origin, i.e. electrical parameters (potential, current, resistance), mechanical quantities (force, pressure, movement), acoustic parameters (language, non-verbal articulations, body sounds), thermal parameters (temperature, body heat) and chemical variables (concentration, pH).

2.1. Electrophysiological multi-channel signals

Research on electrophysiological biosignals, such as recorded electromyography (EMG) and electroencephalography (EEG), usually involves the processing of high dimensional multi-channel time series, such as [12]. Typically, small amounts of data are recorded in settings that differ from experiment to experiment. Therefore, processing pipelines vary from data set to data set and evaluation scripts can be complex (e.g. including parameter optimization by nested cross-validations). In addition to that, research involves careful data screening and statistical analysis.

In BioKIT, researchers can perform user-in-the-loop data analysis using BioKIT's interactive Python shell mode, in which the user can completely control BioKIT, while having the full support of the Python scripting language for easy signal processing, analysis and visualization (e.g. using SciPy tools [13]), as well as debugging.

BioKIT is used for EMG-based Silent Speech Recognition, where the electrical potentials of a user's facial muscles are captured in order to retrace speech. This technology allows to process and recognize speech silently, i.e. by articulating speech without producing an acoustic signal (e.g. when communication in a quiet place is required or for speech-disabled users).

2.2. Motion

We use BioKIT for human motion recognition in gesture and activity detection tasks. With Airwriting [14], we propose a wearable text entry system, in which the user's hand is used as a stylus and text can be written in the air. We are also developing an activity recognition system for an adaptive knee orthosis [15]. The workflow when designing such systems is to first record data for the *Atoms* alone and to build and evaluate a simple multi-class recognizer, where each class relates to one *Atom*. In the above cases this means isolated letters or isolated activity data. In the second step realistic sequence data is obtained and the models obtained in the first step are used to evaluate the sequence recognition abilities. BioKIT streamlines this process by providing a user friendly front-end to build and evaluate simple context-independent classifiers with the option to extend them to the use case of recognizing sequences of patterns. The complexity of context-dependent modeling as used in ASR is hidden from the user.

BioKIT also features modeling of *Atoms* in a hierarchical manner. A complete activity might entail reaching for a bottle, grasping for its lid, and then closing the bottle. Getting the bottle and its lid might appear in arbitrary order but closing the bottle is an action that requires synchronous manipulation with both hands. The closing operation can be modeled as one *Atom* at the highest level, which means the HMM describes the full feature space (e.g. joint angles of both arms), whereas getting the bottle and lid can be modeled at the second hierarchy level, where the action of each arm is modeled by one *Atom*. In this case, the corresponding HMMs only describe the feature space of one arm. This allows for efficient modeling of asynchronous movements of different body parts.

2.3. Acoustic speech

The recognition of acoustic speech, compared to the processing of other biosignals mentioned in this paper, is most concerned with processing large amounts of data, and complex modeling, such as *Atom* context-dependent models. The variability in the data is reflected in the size of the models, requiring toolkits to handle these bigger models [11]. To meet these requirements we parallelize the processing of multiple test sequences utilizing multi-core machines with threads sharing their data as much as possible to minimize overall memory usage. We first generate a prototype configuration of the decoding setup (i.e. loading the required models and creating the search network) and then generate copies from the prototype in memory, limiting model read operations to one. Created copies share thread safe modules. Thread safety is managed by each component individually.

3. Toolkit overview

BioKIT contains documentation including a tutorial as well as extensive commentaries within the code to reduce the effort for new users. A suite of integration and unit tests is included to verify the implemented algorithms. The following sections discuss each component.

3.1. Signal preprocessing

The preprocessing is organized in modules which can be freely combined into preprocessing chains. Each module is able to process any kind of multi-channel data which we represent as lists of matrices where each matrix corresponds to a single channel. Some modules in BioKIT are windowing, spectral analysis, linear transformation (e.g Linear Discriminant Analysis or feature-space Maximum Likelihood Linear Regression (fMLLR) [16]). Common preprocessing modules are implemented in the C++ layer, but new modules can also be implemented directly in Python allowing for rapid prototyping of new preprocessing modules. As all matrices in the Python layer are represented as NumPy arrays, the NumPy and SciPy libraries [13] can be seamlessly integrated. This offers the whole range of mature and optimized signal processing algorithms from these libraries to the user. The preprocessing chains are implemented in Python, enabling the user to modify and extend preprocessing operations on the fly. Processing modules can be reconfigured and exchanged during run-time.

3.2. Scoring feature vectors

The *FeatureVectorScorer* manages the observation models in the HMM states and computes the emission probabilities. *FeatureVectorScorer* can be substituted with any valid implementation of the scorer interface to compute a score for a given model identifier and feature vector. We support Gaussian Mixture Models and Quicknet-trained neural networks[17]. In addition, if a compiled version of Kaldi is found during configuration of BioKIT, wrapper classes integrate a range of Kaldi models [1]. Currently supported adaptation methods are MLLR [16] and Maximum-A-Posteriori adaptation [18].

3.3. Token sequence modeling

The *TokenSequenceModel* (TSM) computes the probability of a *Token* given a sequence of previous *Tokens*. An important reason for us to develop a dynamic decoder is the ability to use arbitrary knowledge sources for modeling token sequences (concerning the type of model itself, as well as supporting very small to very large models). Currently, we support a grammar, a standard n-gram model in the ARPA format [19], a factored language model [20], and the possiblity to interpolate between *TokenSequenceModels*.

3.4. Decoder

The decoder employs a token passing algorithm on a determinized and minimized search network [21]. Context-dependent modeling of *Atoms*, commonly used in ASR (see Section 2.3), is directly integrated into the network with unlimited context size. The network construction's only constraint is to use left-to-right topologies. The search network is a *Token*-loop with each node handling several hypotheses with different *TokenSequenceContexts*. A *TokenSequenceContext* manages the sequence of *Tokens* already recognized as part of a hypothesis and is used as input for the *TokenSequenceModel*.

During the decoding all recognized *Tokens* are compressed to a graph. This lattice can be used for a subsequent n-best list generation or rescoring of hypotheses. The decoder is able to perform a segmentation on the preliminary lattice and return partial results, making it online capable. A segment boundary is found if the duration of a filler *Token* exceeds a given threshold.

Similarly to Soltau et al. [21], we have absolute and relative pruning parameters limiting the number of hypotheses, active HMM states in the search network, as well as the number of HMM states at the end of *Tokens*. By implementing a *TokenSequenceModel* lookahead [22], we can integrate any *TokenSequenceModel* information as soon as possible into the decoding process. A wrapper class implementing the *TokenSequenceModel* interface caches frequent queries employing a least recently used strategy and thus speeds up the search.

3.5. Error blaming

Inspired by the work of Chase [2, 3], we chose to integrate a semi-automatized error identification component directly into the toolkit. The error blaming module collects statistics directly during the decoding, enabling the user to perform the blaming on a partial hypothesis at the state level rather than on the lattice. The tool blames the different components of the system along with model confusions. The tool has proven not only to be helpful for giving feedback on experiments, but also for finding algorithmic errors during development despite a range of unit and integration tests.

Reference-Frames:	73 - 85	86 - 103	104 - 147
Hypothesis-Frames:	73 - 85	86 - 103	104 - 147
Reference:	with(3)	that	surge has(2)
Hypothesis:	with(3)	that	searches
Scorer-Ref:	758.64	995.68	2799.51
Scorer-Hypo:	758.64	995.68	2725.93
TSM-Ref:	5.06	58.48	179.89
TSM-Hypo:	5.06	58.48	136.58
Error-Category:	CORRECT	CORRECT	SCORER_TSM_ERROR

Figure 1: Partial Error Blaming output with error categories and scores for *FeatureVectorScorer* (Scorer) and *TokenSequenceModel* (TSM).

Figure 1 shows the partial output of an error blaming procedure on a single utterance taken from the WSJ0 corpus [23]. The reference and hypothesis are aligned and split into *error regions*. Each region is assigned one of several error categories, with *CORRECT* stating that there is no error in the segment. Along with the error category, information on *FeatureVectorScorer* and *TokenSequenceModel* scores are given. The scores are presented as negative logarithmic values. Figure 1 shows three *error regions*. In the third region both *FeatureVectorScorer* and *TokenSequenceModel* scores favor the erroneous hypothesis over the reference. Due to the acoustic similarity between "surge has" and "searches", the error could be fixed by improving the *TokenSequenceModel*.

4. Experiments

We present the results on three different tasks using BioKIT: Airwriting recognition, EMG-based speech recognition, and automatic acoustic speech recognition.

4.1. Emg-based speech recognition

The EMG system consists of a *multi-stream* setup with 8 ("streams") [24]. These streams correspond to *phonetic features* such as the place or manner of articulation; furthermore *phonetic feature bundling* [24] is performed to model dependencies between phonetic features. Each stream is modeled with GMMs. On a 108-word vocabulary, we obtain on average

20.88% word error rate on session-dependent systems, where training and test data were jointly recorded. Experiments were conducted on the EMG-UKA corpus, the data set contains 12 sessions on 4 speakers [25].

4.2. Airwriting recognition

In airwriting, users write text in the air as they were writing on a virtual notepad or blackboard. The handwriting motion is measured by an accelerometer and a gyroscope attached to the back of the hand. Text is written in uppercase block letters. For the airwriting task, we use a context independent system with 26 atoms representing the uppercase letters of the latin alphabet [14]. Each atom is modeled with a 30-state left-to-right HMM. The system is initialized on single letter recordings and additional training is performed on single word recordings. On a corpus with 9 subjects, each contributing 80 written sentences, we obtain on average 11% word error rate in a user-independent setup (leave-one-out cross-validation) with a vocabulary of 8000 words and a 3-gram language model. The word error rate drops to 3% for the user-dependent case.

4.3. Automatic speech recognition

We report on experimental results conducted with our decoder and compare the results to Kaldi [1]. Furthermore, we investigate the real-time factor necessary to obtain them. We chose Kaldi for our comparison since it offers state-of-the-art acoustic modeling and performance, and is freely available.

4.3.1. Experimental setup

We report results on the Bulgarian (BG), Czech (CZ), German (GE), Mandarin (CH), and Vietnamese (VN) part of the GlobalPhone database [26]. Each system is trained on about 22 hours of read speech with the Kaldi toolkit. We present the results obtained on deep neural network (DNN) acoustic models from [27] using an fMLLR adaptation. The fMLLR transforms were separately created by Kaldi and BioKIT in a first pass by using Kaldi trained Gaussian Mixture Models. The input for the DNN is a 143 dimensional feature vector consisting of 11 stacked 13 dimensional MFCC vectors. The DNN is initialized by training on a range of languages from the GlobalPhone database and is then re-trained with the target language training data. The neural network has 5 hidden layers with 1,500 nodes each.

The dictionary sizes and language model (LM) information are given in Table 1. The number of highest order n-grams are presented in the table to give an estimate of the size of the model. The appended "s" and "b" indicate small and big language models, respectively. The out-of-vocabulary (OOV) rate for each system is less than 0.01% with the exception of Bulgarian where we have an OOV of 2.18%. For the experiments with BioKIT, the full language model lookahead was used for each system with an unlimited lookahead depth.

In order to limit network size and reduce memory consumption, Kaldi integrates big language models on the fly using the network build on the small LM. To be able to use DNNs in conjunction with our big LMs in Kaldi, we adapted the Kaldi decoder script.

4.3.2. Results

First, we report the error rates of our systems using the Kaldi and the BioKIT decoder in Table 1. The results for Mandarin are given in character error rate (CER), Vietnamese in syllable error rate (SER), and Bulgarian, Czech, and German in word error rate (WER). The results show that the Kaldi and BioKIT decoder achieve comparable error rates for all systems. Results of BioKIT and Kaldi systems were tested pair-wise for significance. Differences between systems were not significant at a signficance level of 0.05 except for BGs and CH systems.

Table 1: *Various systems and their performance with Kaldi and BioKIT given in WER. Mandarin is given in CER and Vietnamese in SER. All systems use a 3-gram LM with the exception of VNb which employs a 5-gram LM.*

system	words	PPL	n-grams	BioKIT	Kaldi
BGs	100k	386	1.7m	12.50%	12.84%
BGb	100k	306	47.3m	11.76%	12.16%
CZs	33k	1,644	4.3m	9.19%	9.23%
CZb	33k	1,469	15.3m	8.73%	8.66%
GEs	37k	673	2.2m	10.89%	10.85%
GEb	37k	552	33.3m	9.50%	9.76%
CH	71k	503	5.0m	17.14%	16.90%
VNs	30k	247	1.7m	8.17%	8.10%
VNb	30k	179	50.6m	6.99%	7.10%

Finally, Figure 2 shows BioKIT's performance in terms of real-time factor over syllable error rate on our Vietnamese systems. The decoding speed of our decoder is not affected by the bigger language model. The decrease in error rate levels off after a real-time factor of about 1. Tests were run on an Intel Core i7-3770 with 3.4GHz and 4 cores. All tests were run with 4 threads in parallel. Real-time factor is derived from the sum of the decoding times of all 4 threads. For the Vietnamese systems, the memory consumption for experiments with a single thread is 1.1GB (VNs) and 2.4GB (VNb). Using our parallelization with 4 threads, memory consumption increases to 3.3GB (VNs) and 4.6GB (VNb).

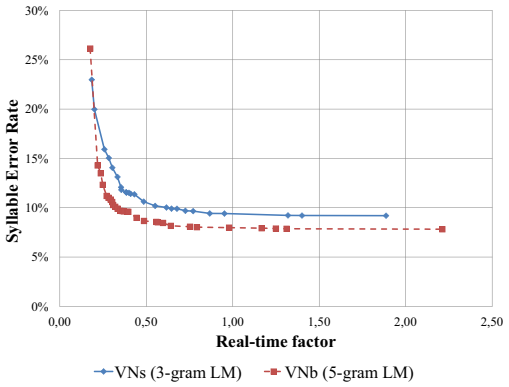

Figure 2: Real-time factor over SER for GlobalPhone Vietnamese system with 3-gram LM(VNs) and 5-gram LM(VNb).

5. Conclusion

We introduced our new toolkit BioKIT and gave an overview of its components. We showed that our toolkit is suitable for several human-machine interfaces applying a variety of different biosignals. We reported on first results of our decoder using Kaldi trained deep neural networks achieving comparable results. We showed that usage of large language models has no negative effect on real-time factor over error rate with BioKIT. Furthermore, by employing our parallelization strategy of sharing modules between threads, we are able to reduce the memory requirements significantly. Future plans include extending the error blaming component by increasing the granularity of error categories, proposing likely solutions to frequent errors automatically, and enabling automatic tuning of a given decoding setup.

6. References

[1] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, and P. Schwarz, "The Kaldi Speech Recognition Toolkit," in *IEEE Automatic Speech Recognition and Understanding (ASRU) Workshop*, 2011, pp. 1–4.

[2] L. L. Chase, "Error-Responsive Feedback Mechanisms for Speech Recognizers," Ph.D. dissertation, Pittsburgh, PA: Carnegie Mellon University, 1997.

[3] H. Nanjo, A. Ri, and T. Kawahara, "Automatic Diagnosis of Recognition Errors in Large Vocabulary Continuous Speech Recognition System," *Joho Shori Gakkai Kenkyu Hokoku*, vol. 99, no. 64, pp. 41–48, 1999.

[4] S. J. Young, G. Evermann, M. Gales, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, "The HTK Book Version 3.4," 2006.

[5] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel, "Sphinx-4: A Flexible Open Source Framework for Speech Recognition," *Sun Microsystems Technical Report*, 2004.

[6] A. Lee, T. Kawahara, and K. Shikano, "Julius - An Open Source Real-Time Large Vocabulary Recognition Engine," in *Eurospeech*, 2001, pp. 1691–1694.

[7] D. Rybach, C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, and H. Ney, "The RWTH Aachen University Open Source Speech Recognition System." in *Interspeech*, 2009, pp. 2111–2114.

[8] T. Westeyn, H. Brashear, A. Atrash, and T. Starner, "Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition," in *International Conference on Multimodal Interfaces (ICMI)*, 2003, pp. 85–92.

[9] M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries, and M. Westphal, "The Karlsruhe-Verbmobil Speech Recognition Engine," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 1997, pp. 83–86.

[10] H. Soltau, F. Metze, C. Fügen, and A. Waibel, "A One-Pass Decoder Based on Polymorphic Linguistic Context Assignment," in *IEEE Automatic Speech Recognition and Understanding (ASRU) Workshop*, 2001, pp. 214–217.

[11] H. Soltau, G. Saon, and B. Kingsbury, "The IBM Attila Speech Recognition Toolkit," in *IEEE Workshop on Spoken Language Technology (SLT)*, 2010, pp. 97–102.

[12] M. Wand, C. Schulte, M. Janke, and T. Schultz, "Array-based Electromyographic Silent Speech Interface." in *BIOSIGNALS*, 2013, pp. 89–96.

[13] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001. [Online]. Available: http://www.scipy.org/

[14] C. Amma, M. Georgi, and T. Schulz, "Airwriting: A Wearable Handwriting Recognition System," *Personal and Ubiquitious Computing*, pp. 1–13, 2013.

[15] D. Rebelo, C. Amma, H. Gamboa, and T. Schultz, "Human Activity Recognition for an Intelligent Knee Orthosis," in *International Joint Conference on Biomedical Engineering Systems and Technologies - Biosignals*, 2013, pp. 368–371.

[16] M. J. Gales, "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition," *Computer Speech & Language*, vol. 12, no. 2, pp. 75–98, 1998.

[17] D. Johnson, D. Ellis, C. Oei, C. Wooters, and P. Faerber, "Quicknet," 2005. [Online]. Available: http://www.icsi.berkeley.edu/Speech/qn.html

[18] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *Speech and audio processing, ieee transactions on*, vol. 2, no. 2, pp. 291–298, 1994.

[19] A. Stolcke, "SRILM An Extensible Language Modeling Toolkit," in *Interspeech*, 2002, pp. 901–904.

[20] J. A. Bilmes and K. Kirchhoff, "Factored Language Models and Generalized Parallel Backoff," in *HLT-NAACL*, vol. 2, 2003, pp. 4–6.

[21] H. Soltau and G. Saon, "Dynamic Network Decoding Revisited," in *IEEE Automatic Speech Recognition and Understanding (ASRU) Workshop*, 2009, pp. 276–281.

[22] S. Ortmanns, A. Eiden, H. Ney, and N. Coenen, "Look-Ahead Techniques for Fast Beam Search," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 3, 1997, pp. 1783–1786.

[23] J. Garofalo, D. Graff, D. Paul, and D. Pallett, "Continous Speech Recognition (CSR-I) Wall Street Journal (WSJ0) News, Complete," Linguistic Data Consortium, Philadelphia, Tech. Rep., 1993.

[24] T. Schultz and M. Wand, "Modeling Coarticulation in Large Vocabulary EMG-based Speech Recognition," *Speech Communication*, vol. 52, no. 4, pp. 341 – 353, 2010.

[25] M. Wand, M. Janke, and T. Schultz, "The EMG-UKA Corpus for EMG-based Speech Recognition." in *Interspeech*, 2014.

[26] T. Schultz, N. T. Vu, and T. Schlippe, "GlobalPhone: A Multilingual Text & Speech Database in 20 Languages," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

[27] N. T. Vu, D. Imseng, D. Povey, P. Motlicek, T. Schultz, and H. Bourlard, "Multilingual Deep Neural Network based Acoustic Modeling For Rapid Language Adaptation," in *in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.