ITERATIVE BAYESIAN WORD SEGMENTATION FOR UNSUPERVISED VOCABULARY DISCOVERY FROM PHONEME LATTICES

Jahn Heymann, Oliver Walter, Reinhold Haeb-Umbach[*]

University of Paderborn
Department of Communications Engineering
Paderborn, Germany

Bhiksha Raj [†]

Carnegie Mellon University
Language Technologies Institute
5000 Forbes Avenue, Pittsburgh
PA 15213, United States

ABSTRACT

In this paper we present an algorithm for the unsupervised segmentation of a lattice produced by a phoneme recognizer into words. Using a lattice rather than a single phoneme string accounts for the uncertainty of the recognizer about the true label sequence. An example application is the discovery of lexical units from the output of an error-prone phoneme recognizer in a zero-resource setting, where neither the lexicon nor the language model (LM) is known. We propose a computationally efficient iterative approach, which alternates between the following two steps: First, the most probable string is extracted from the lattice using a phoneme LM learned on the segmentation result of the previous iteration. Second, word segmentation is performed on the extracted string using a word and phoneme LM which is learned alongside the new segmentation. We present results on lattices produced by a phoneme recognizer on the WSJ-CAM0 dataset. We show that our approach delivers superior segmentation performance than an earlier approach found in the literature, in particular for higher-order language models.

Index Terms— Automatic speech recognition, Unsupervised learning, Word Segmentation

1. INTRODUCTION

Conventional speech recognition systems rely on supervised learning, where the patterns to be recognized (the words) and their acoustic building blocks (the phonemes) are known a priori and where labeled training data are used to learn the models. In unsupervised vocabulary discovery, however, the word inventory is not known in advance and the training data come without labels. There are several applications where such a problem arises, such as the development of an automatic speech recognition (ASR) system for languages for which low or zero resources are available. Another application is teaching a robot an environment-specific vobabulary by speech interaction. Further, the techniques can serve as a computational model for early child language acquisition.

Unsupervised word discovery is essentially based on searching for recurring sequential patterns in the audio data. The algorithms, however, should allow for some variation to account for the well-known inter- and intra-speaker variability in the acoustic realization of the same linguistic phrase. Several algorithms have been proposed for this task. Segmental dynamic time warping has been used for discovering repeated acoustic patterns in the input speech [1]. Others rely on a two-stage approach, where first the basic acoustic building blocks, i.e. the phonemes or similar entities, are discovered and models are trained for them [2]. Once this has been achieved, the real-valued speech input can be transcribed into sequences of symbols, the phoneme labels. The second task is then the discovery of lexical units, i.e., the words, as recurring sequences of these symbols. This second task is the objective of the paper at hand.

The discovery of word-like lexical units can be achieved by using a HMM approach [3, 4, 5, 6], while others have suggested a multigram model [7] or applied non-negative matrix factorization [8]. An elegant method that overcomes many of the shortcomings of the aforementioned techniques is based on unsupervised nonparametric Bayesian word segmentation: The lexicon size need not be specified in advance, as in the HMM approach of [5], it does not require the tuning of a threshold, as with the multigram approach [7], and it employs n-gram statistics rather than bag-of-units or co-occurrence counts as in the NMF approach of [8].

Mochihashi et al. used a nested Pitman-Yor language model [9] to segment a character input into words [10]. Their algorithm can be immediately applied to segment an error-free phoneme sequence into words. However, it is unable to cope with errors in the input. Neubig et al. extended the approach to noisy input, represented by a phoneme lattice, employing Weighted Finite State Transducers (WFSTs) [11]. However, for an exact implementation, the computational complexity becomes intractable for language models of order larger than unigram [12]. In this contribution we therefore investigate a suboptimal, iterative approach, which alternates between determining the most probable phoneme sequence given the current estimate of the language model and carrying out language model estimation and word segmentation on that phoneme sequence. While first results on artificially noisified character lattices have been presented in [12], several extensions are presented here to achieve competitive results on real phoneme lattices produced by an ASR engine. In the results section we will show that this approach delivers similar word discovery results as the one of [11] for unigram language models while it is significantly better for bigram language models. We will also discuss the impact of the density of the phoneme lattice on the performance of word discovery for different language model orders.

2. WORD DISCOVERY FROM PHONEME INPUT

The objective function of speech recognition is the posterior probability of the word sequence W given the acoustic input X. Usually a language model, which consists of a word list and word (sequence) probabilities is employed to support the recognition. We make this

*Supported by Deutsche Forschungsgemeinschaft under contract no. Ha 3455/9-1 within the Priority Program SPP1527 "Autonomous Learning".
†Supported by NSF grant 1017256.

knowledge explicit in our notation by conditioning the posterior on the language model (LM) estimate $\hat{G}$:

$$\hat{W} = \underset{W}{\operatorname{argmax}} \mathrm{P}\left(W|X, \hat{G}\right) \\ = \underset{W}{\operatorname{argmax}} \mathrm{p}\left(X|W, \hat{G}\right) \mathrm{P}\left(W|\hat{G}\right). \quad (1)$$

Here, $\mathrm{p}(X|W, \hat{G})$ is the acoustic model likelihood, while the lexicon probability $\mathrm{P}(W|\hat{G})$ is computed using the LM.

In large vocabulary speech recognition the acoustic models of words are obtained by concatenating the acoustic models of phonemes according to a pronunciation dictionary. Thus the phonemes are introduced as a hidden variable [11]:

$$\hat{W} = \underset{W}{\operatorname{argmax}} \sum_{Y} \mathrm{p}(X|Y) \mathrm{P}\left(Y|W, \hat{G}\right) \mathrm{P}\left(W|\hat{G}\right). \quad (2)$$

Here $\mathrm{p}(X|Y)$ is the acoustic model probability for phonemes Y, while $\mathrm{P}(Y|W, \hat{G})$ is a pronunciation lexicon probability, which is nonzero only if the concatenation of the phonemes Y gives the hypothesized word sequence W. The above optimization is usually approximated by searching for the most likely phoneme sequence:

$$\hat{W} \approx \underset{W,Y}{\operatorname{argmax}} \mathrm{p}(X|Y) \mathrm{P}\left(Y|W, \hat{G}\right) \mathrm{P}\left(W|\hat{G}\right). \quad (3)$$

We now turn to the estimation of the LM G. In the zero resource setting considered here it must be estimated on the very same acoustic data, whose recognition it is supposed to support. In our approach it is estimated in a Bayesian manner from the corpus $\mathcal{W}$ of phoneme strings that have been segmented into words:

$$\hat{G} = \underset{G}{\operatorname{argmax}} \mathrm{P}(G|\mathcal{W}) \\ = \underset{G}{\operatorname{argmax}} \mathrm{P}(\mathcal{W}|G) \mathrm{P}(G). \quad (4)$$

$\mathrm{P}(\mathcal{W}|G)$ is the word likelihood given the LM probabilities and $\mathrm{P}(G)$ is the prior for the LM. As the LM G we use the *Nested Pitman-Yor Language Model* (NPYLM) and set the prior accordingly. The NPYLM is briefly described in Section 3.

Looking at eqs. (3) and (4), we see that for the search for the most probable word sequence, the language model estimate $\hat{G}$ is required, while the estimation of $\hat{G}$ requires the transcription of the input data in terms of segmented phoneme sequences. To solve this, we propose an iterative approach, which is described in Section 4.

3. NESTED PITMAN-YOR LANGUAGE MODEL

In Bayesian language modeling an a priori probability $P(G)$ of the language model is required, see eq (4). For this purpose the Pitman-Yor process has been proposed, which produces power-law distributions which more closely resemble the statistics found in natural languages than the probabilities produced by Dirichlet distributions [13]. An n-gram language model $G_{w_{i-n+1}^{i-1}}$ is a multinomial distribution of probabilities for the N words of the vocabulary: $G_{w_{i-n+1}^{i-1}} = \{g_{w_i=1|w_{i-n+1}^{i-1}}, \ldots, g_{w_i=N|w_{i-n+1}^{i-1}}\}$. Here, the subscript denotes the context $u = w_{i-n+1}^{i-1}$, i.e., the preceding $n-1$ words. In the hierarchical Pitman-Yor process $G_{w_{i-n+1}^{i-1}}$ is modeled as a draw

$$G_{w_{i-n+1}^{i-1}} \sim PY(d_n, \theta_n, G_{w_{i-n+2}^{i-1}}) \quad (5)$$

from a Pitman-Yor process with discount parameter d_n, strength parameter θ_n and base measure $G_{w_{i-n+2}^{i-1}}$ [9]. While the discussion of the first two parameters is beyond the scope of this paper it is important to note that the base measure, which corresponds to the expected probability distribution of the draws, is set to the language model $G_{w_{i-n+2}^{i-1}}$ of the parent $(n-1)$-gram. This process is repeated until the parent LM is a zerogram. Since in word segmentation the vocabulary size is not known in advance the zerogram cannot be specified. It is therefore replaced by the likelihood for the word being a phoneme sequence calculated by a phoneme language model H', where again a hierarchy of phoneme language models is built up to some order m. The resulting structure is the NPYLM G, which consists of a hierarchical Pitman-Yor LM (HPYLM) for words and a HPYLM for phonemes, the latter denoted by H'.

4. ITERATIVE OPTIMIZATION

To solve the chicken-and-egg problem outlined in Section 2 we have proposed an iterative approach in [12]. While this delivered good vocabulary discovery results on input character sequences that have been artificially noisified by adding character alternatives to the lattice drawn uniformly at random [12], the performance obtained on real phoneme lattices produced by an ASR engine turned out to be disappointing. The reason is probably that the phoneme errors of a recognizer are by no means uniformly distributed.

In the following we therefore propose two modifcations which produced significantly better results on real lattices. The first modification is to use separate language models for phoneme recognition and for word segmentation. The latter is the NPYLM G, which consists of HPYLMs for words and phonemes, which are nested, while the former is a HPYLM H for phonemes only. The second modification is to increase the language model order only after k_{sw} iterations instead of starting with a high-order LM upfront. The following algorithm summarizes the proposed iterative approach to vocabulary discovery:

Algorithm 1 Iterative vocabulary discovery from raw speech

```
**Input:** $\mathcal{X}, k_{\text{sw}}$
**Output:** $\mathcal{Y}, \mathcal{W}, G, H$
Initialization: Set $G, H$ to phoneme zerograms, $k = 1$
**while** $k \leq k_{\max}$ **do**
    1) Transcribe each speech utterance $X$ into phoneme sequence $Y$ using HPYLM $H$, resulting in a corpus $\mathcal{Y}$ of phoneme strings: $\mathcal{X} \overset{H}{\rightarrow} \mathcal{Y}$
    2a) Carry out word segmentation on the phoneme strings, using the NPYLM $G$, resulting in a corpus $\mathcal{W}$ of phoneme strings segmented into words: $\mathcal{Y} \overset{G}{\rightarrow} \mathcal{W}$
    2b) Re-estimate the NPYLM language model $G$ and the HPYLM phoneme language model $H$: $\mathcal{W} \rightarrow G, H$
    **if** $k = k_{\text{sw}}$ **then**
        Increase language model orders
    **end if**
    $k = k + 1$
**end while**
```

The first step in the repeat loop is carried out by a phoneme recognizer. However, to save the computational effort of repeated phoneme recognition, a phoneme lattice is produced by the ASR engine in the first iteration, and the updated HPYLM H is applied by rescoring in later iterations. Then the most probable phoneme string is extracted from the lattice using Viterbi decoding. Tasks 2a) and 2b, i.e., word segmentation and language model estimation, are carried out on the currently most probable phoneme sequence using the algorithm of [10].

The motivation for using different LMs for phoneme recognition and word segmentation is twofold. First, the phoneme recognizer should use the best phoneme LM that is available. The NPYLM G is a nested LM which contains probabilities of words and of phonemes, while the latter has been estimated on the parts of the input phoneme sequence generated by the base distribution [10]. The phoneme LM part of G is thus only trained on a small subset of the overall corpus. Furthermore, this subset will most likely not represent the statistics of the whole corpus well. On the other hand H is a pure phoneme LM trained on the whole segmented corpus $\mathcal{Y}$. It should thus be more powerful. Second, we observed in the experiments that while a high model order n of the phoneme n-gram LM is beneficial for ASR, it is not so for word segmentation: here, better segmentation results were achieved if the phoneme LM part of G had a low order in the initial iterations. The reason for the worse performance of word segmentation using high-order n-grams is probably that, due to the errors in the input phoneme strings, not much consistency can be found in long sequences of phonemes.

It is important to note that the phoneme LM H contains a word end tag, whose probability is trained along with the phoneme probabilities. By doing so, some information about the word segmentation is exploited in the phoneme recognition, which turns out to be quite beneficial.

Successively increasing the LM orders can be motivated as follows. Higher-order LMs deliver better segmentation results than low-order LMs if the input sequence is noisefree [12]. On the other hand initialization of higher-order LMs from noisy input is more difficult and is likely to lead to convergence to a local optimum. Starting with low orders and increasing the orders apparently leads to convergence to a better optimum.

5. WFST BASED IMPLEMENTATION

As proposed in [11] we use a WFST for calculating the probability of every possible segmentation for an input sentence. The WFST itself is composed of two WFSTs: The first is the lexicon WFST, which stores already discovered words and also all possible yet unseen words that are contained in the presented input sentences. This WFST is responsible for generating all possible segmentations into known and unknown words. A recurring phoneme sequence is transduced into a known word, a phoneme sequence observed for the first time is considered an unknown word and is passed through, only adding a word end tag after the last phoneme. To generate this lexicon WFST our algorithm analyses the phonetic transcriptions of all utterances of the corpus to decide if a phoneme sequence is a previously seen sequence and thus a known word or not. This is done to avoid hypothesizing already known words as unknown words again, as this would lead to falsely calculated probabilites for a given phoneme sequence. This is in contrast to [11], where the possible segmentation contains both variants: one with the phonetic sequence replaced by the known word and one with the same sequence as an unknown, newly hypothesized word.

If the input phoneme string of an utterance were replaced by a phoneme lattice to allow for alternative transcriptions, the analysis for known words would for all but toy lattices quickly become overwhelmingly large rendering the approach computationally intractable. For this reason the analysis is only carried out on the most probable phoneme sequence.

The approach in [11] does not face this computational issue. However, it leads to wrongly computed probabilities for word LMs in G higher than unigram, as was discussed in [12].

The second WFST is the language model and it is designed in a similar way as in ASR as a weighted finite state acceptor. It is used to calculate the likelihood for every possible segmentation when composing it with the lattice produced by composing the input lattice and the lexicon. The language model WFST is designed to represent the structure of the NPYLM and calculates the probabilities accordingly. The structure is the same as proposed in [11].

6. EXPERIMENTAL RESULTS

For the experiments a monophone HMM acoustic model was trained using the WSJCAM0 training set comprising 7861 sentences. Decoding was carried out on a subset of the WSJCAM0 training set, which consisted of 5628 sentences containing about 10 hours of audio, using the monophone models and a zerogram phoneme LM, producing a lattice at the output. We chose to use the training data because it contains more data than the test and evaluation sets. In our view this does not lead to overly optimistic results compared to a test on independent data, because about the same phoneme error rate of 33% was achieved as on the test and evaluation data. The size of the vocabulary is about 10k words.

For evaluation we used the word segmentation token F-score, where a higher F-score indicates better word discovery performance. The number of correct words in the segmented phoneme sequence is determined by aligning the segmented words and the reference string according to the minimum Levenshtein distance.

To better assess the performance of the segmentation algorithms, lower and upper bounds for the F-score were determined as follows.

To arrive at a lower bound the single best path was extracted from the phoneme lattice and the segmentation algorithms were run on this resulting phoneme string. This corresponds to the segmentation result obtained when only doing step 2a) and 2b) of the algorithm 1, where the phoneme recognition was based on a zerogram phoneme LM. Clearly, if the proposed iterative optimization on the lattice is of any use, it should deliver a higher F-score than this lower bound.

To obtain an upper bound on the F-score, the path with the minimal edit distance to the ground-truth transcription was extracted from the lattice. Its phoneme error rate is called *lattice phoneme error rate* (L-PER) in the following. The word segmentation performance on this phoneme string is the best that can possibly be achieved by algorithm 1. No better performance is possible for the given input lattice, as no phoneme string with a smaller phoneme error rate is present in the given lattice produced by the ASR decoder.

To study the impact of the L-PER on the word segmentation performance lattices of different size were generated by chopping off all paths of an initially dense lattice, whose likelihood was $\exp(X)$ times worse than the likelihood of the best path, were X ranged from 1 to 10 with a stepsize of 1. By doing so, lattices with different L-PER were generated with the smallest L-PER for the most dense lattice.

To further set the results into perspective we compared the proposed algorithm with the algorithm proposed in [11] using the freely available software published with the paper [14].

Table 1 summarizes the different setups. We ran all algorithms for $k_{\text{max}} = 100$ iterations. After $k_{\text{sw}} = 35$ iterations the switch to the higher-order LMs was performed for setup (1). The resulting F-scores are shown in Fig. 1 as a function of the L-PER of the input lattice for different configurations.

Table 1. Order of the LMs for the different setups

	G	H'	H	algorithm
● (1)	$1 \rightarrow 2$	$2 \rightarrow 8$	$4 \rightarrow 8$	proposed
▼ (2)	1	2	4	proposed
★ (3)	1	2	-	[11, 14]
◆ (4)	2	8	-	[11, 14]
(5)	1	2	-	[12]
(6)	2	8	8	proposed

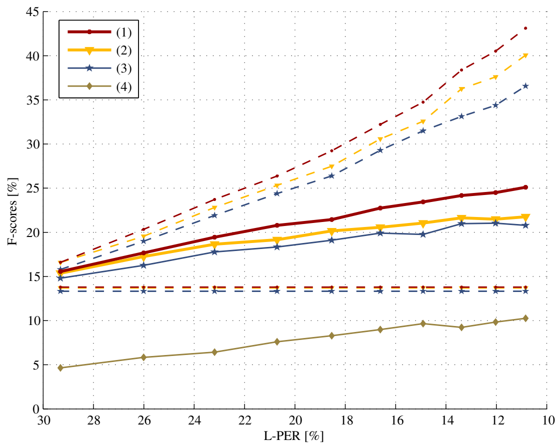

Fig. 1. F-score over L-PER for different setups, see Table 1 for explanation. The dashed lines show the bounds for each configuration.

It can be observed that the F-score increases with decreasing L-PER for both the proposed and reference algorithm. Both algorithms are able to extract a path from the lattice with a lower phoneme error rate than the initial single best path, which was obtained from a zerogram phoneme LM. Obviously, the LM estimated on the error-prone phoneme output is indeed helpful as it is able to improve the phoneme recognition of the ASR decoder. However, the gain obtained with the proposed algorithm is larger than the one obtained with [11], in particular with a bigram word LM in G.

The better bigram result is in contrast to the results reported in [15], where in case of noisy input an unigram word model performed better than a bigram word model. Our approach therefore seems to be more robust against noisy input. The reason for this might be the fact that the bigram word model is initialized with the unigram segmentation result and further in the structure of the NPYLM with an effective smoothing and fallback strategy. As the F-score for the bigram model is the highest, our approach seems promising because with error free text higher order LMs have been shown to significantly outperform lower order LMs [16]. We expect the same thing to happen when the lattices have a lower PER on the single best path.

Figure 2 shows the values of the performance measures over the course of the iterations for our proposed algorithm. It can clearly be seen that the switch after $k_{\text{sw}} = 35$ iterations improves the results almost immediately. Without switching the performance would not increase significantly after the 30th iteration. Experiments also showed that starting with the higher-order LMs from the first iteration led to significantly lower performance.

Table 2 summarizes the final results for the different algorithms and setups for the lattice with the lowest L-PER. Additionally, we present the word error rate (WER) on the segmented result. It can clearly be seen that our approach delivers the best performance w.r.t. nearly all performance measures with setup (1). Only the PER is better when starting with a high order LM (setup (6)). This is followed in terms of the F-score, recall and WER when using setup (2). While the approach of [11] with setup (4) delivers a low phoneme error rate, it unfortunately delivers a low word segmentation performance compared to the setup (3). The table also contains the results of our earlier algorithm (5) [12], which have been significantly improved with the techniques described in this paper. Comparing setup (6) and (1) shows that starting with higher order, results in worse performance compared to model order switching.

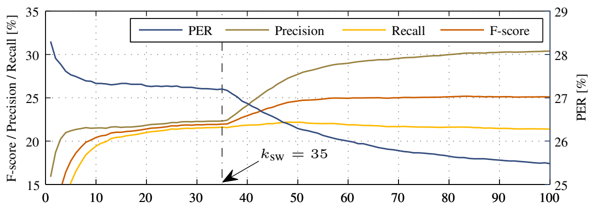

Fig. 2. Performance measures over iterations for setup (1)

Table 2. Results summary for L-PER ≈ 10.5%: legend, see Fig. 1

	● (1)	▼ (2)	★ (3)	◆ (4)	(5)	(6)
F-score	**25.1**	21.8	20.8	10.2	13.9	18.2
Recall	**21.4**	21.4	20.3	15.3	12.1	14.2
Precision	**30.4**	22.1	21.7	7.7	16.5	25.1
PER	25.5	26.9	28.4	26.8	32.4	**24.7**
WER	**79.8**	86.2	91.5	183.9	89.7	86.2

Finally we counted the number of correct words in the lexicon in the final result for setup (1). We achieve a lexicon precision of 13.6%, a lexicon recall of 19.3% resulting in a lexicon F-score of 16.0%. Out of the 100 most often found words, 70% were correct words. Our implementation needed an average of 150 seconds per iteration with the lower model orders and 450 seconds for the higher model orders, resulting in a runtime of about 10 hours for the 100 iterations on a single core of an Intel(R) Xeon(R) E5-2640 at 2.50GHz. Therefore we are able to process the audio data with a realtime factor of one.

7. CONCLUSIONS

We have presented an approach to unsupervised word segmentation on phoneme lattices produced by an ASR decoder. Using an iterative approach which alternates between extraction of the best phoneme sequence from the lattice and word segmentation, introducing a separate phoneme language model for phoneme lattice rescoring, and increasing the model orders only after several iterations have been performed using a low-order LM for word segmentation in the initial iterations, we were able to significantly improve the results compared to our former version [12]. It is also shown to be superior to an existing approach from the literature, in particular when using bigram word language models, while at the same time staying computationally tractable. This work is inspired by the work of Neubig *et al.* [11] and based on the work of Mochihashi *et al.* [10] who extended the work of Teh *et al.* to the unsupervised case. It contributes to the research of zero-resource speech technologies adding to the works on unsupervised word discovery summarized in [15].

8. REFERENCES

[1] Alex S. Park and James R. Glass, "Unsupervised pattern discovery in speech," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 1, 2008.

[2] Sourish Chaudhuri, Mark Harvilla, and Bhiksha Raj, "Unsupervised learning of acoustic unit descriptors for audio content representation and classification," in *Proc. of Interspeech*, 2011.

[3] Aren Jansen and Kenneth Church, "Towards unsupervised training of speaker independent acoustic models," in *Proc. of Interspeech*, 2011.

[4] Meng Sun and Hugo Van Hamme, "Joint training of non-negative Tucker decomposition and discrete density hidden Markov models," *Computer Speech & Language*, vol. 27, no. 4, pp. 969–988, 2013.

[5] Oliver Walter, Timo Korthals, Reinhold Haeb-Umbach, and Bhiksha Raj, "A Hierarchical System For Word Discovery Exploiting DTW-Based Initialization," in *Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec. 2013.

[6] Sourish Chaudhuri and Bhiksha Raj, "Unsupervised Structure Discovery for Semantic Analysis of Audio," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1187–1195.

[7] Sabine Deligne and Frédéric Bimbot, "Inference of variable-length linguistic and acoustic units by multigrams," *Speech Communication*, vol. 23, no. 3, pp. 223–241, 1997.

[8] Veronique Stouten, Kris Demuynck, and Hugo Van Hamme, "Discovering Phone Patterns in Spoken Utterances by Non-Negative Matrix Factorization," *Signal Processing Letters, IEEE*, vol. 15, pp. 131–134, 2008.

[9] Yee Whye Teh, "A hierarchical Bayesian language model based on Pitman-Yor processes," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2006.

[10] Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda, "Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, 2009.

[11] Graham Neubig, Masato Mimura, and Tatsuya Kawaharak, "Bayesian learning of a language model from continuous speech," *IEICE TRANSACTIONS on Information and Systems*, vol. 95, no. 2, 2012.

[12] Jahn Heymann, Oliver Walter, Reinhold Haeb-Umbach, and Bhiksha Raj, "Unsupervised Word Segmentation from Noisy Input," in *Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec. 2013.

[13] Christopher D. Manning and Hinrich Schütze, *Foundations of statistical natural language processing*, MIT press, 1999.

[14] Graham Neubig, *latticelm*, Available at http://github.com/neubig/latticelm, accessed Apr. 2013, v0.2.

[15] Aren Jansen, Emmanuel Dupoux, Sharon Goldwater, Mark Johnson, Sanjeev Khudanpur, Kenneth Church, Naomi Feldman, Hynek Hermansky, Florian Metze, Richard Rose, Mike Seltzer, Pascal Clark, Ian McGraw, Balakrishnan Varadarajan, Erin Bennett, Benjamin Börschinger, Justin Chiu, Ewan Dunbar, Abdellah Fourtassi, David Harwath, Chia-ying Lee, Keith Levin, Atta Norouzian, Vijayaditya Peddinti, Rachael Richardson, Thomas Schatz, and Samuel Thomas, "A summary of the 2012 JHU CLSP workshop on Zero Resource speech technologies and models of early language acquisition," in *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing*, 2013.

[16] Oliver Walter, Reinhold Haeb-Umbach, Sourish Chaudhuri, and Bhiksha Raj, "Unsupervised Word Discovery from Phonetic Input Using Nested Pitman-Yor Language Modeling," ICRA Workshop on Autonomous Learning, 2013.