LEARNING A SUBWORD VOCABULARY BASED ON UNIGRAM LIKELIHOOD

Matti Varjokallio, Mikko Kurimo

Aalto University
Dept. of Signal Processing and Acoustics
Espoo, Finland
{matti.varjokallio,mikko.kurimo}@aalto.fi

Sami Virpioja

Aalto University
Dept. of Information and Computer Science
Espoo, Finland
sami.virpioja@aalto.fi

ABSTRACT

Using words as vocabulary units for tasks like speech recognition is infeasible for many morphologically rich languages, including Finnish. Thus, subword units are commonly used for language modeling. This work presents a novel algorithm for creating a subword vocabulary, based on the unigram likelihood of a text corpus. The method is evaluated with entropy measure and a Finnish LVCSR task. Unigram entropy of the text corpus is shown to be a good indicator for the quality of higher order n-gram models, also resulting in high speech recognition accuracy.

Index Terms— Large Vocabulary Continuous Speech Recognition, Vocabulary Selection, Subword Modeling

1. INTRODUCTION

Morphologically rich languages pose special challenges for natural language processing tasks such as speech recognition and machine translation [1]. Because of morphological processes like agglutination, compounding and inflection, the amount of different word forms may be huge and cause problems for traditional word-based language modeling approach. A common solution is to use a vocabulary consisting of subwords instead of words. Subword-based approaches are widely applied in automatic speech recognition for languages such as Finnish, Estonian, Turkish, Hungarian, Thai, Czech, and Slovenian. For a comprehensive survey of different methods, see [2]. The subword vocabularies selected using unsupervised machine learning methods have been shown to perform well [3, 4], so neither a morphological analyzer nor an annotated training corpus is required. One common unsupervised method is Morfessor [5], which is used as a baseline method in this work.

Modern speech recognizers aimed for large vocabulary tasks typically utilize n-grams for language modeling. To avoid data sparseness issues and achieve good n-gram estimates, the vocabulary size and units should be selected carefully. Also out-of-vocabulary (OOV) rate has to be considered. For a morphologically rich language, using words as vocabulary units leads either to a high OOV rate if the vocabulary is too small, or difficulties in estimating the n-gram probabilities if the vocabulary is too large. By using a subword vocabulary, it is possible to achieve virtually unlimited word vocabulary by using subwords as building blocks. Recognition rates for words not in the training data have been analyzed in [2, 6]. For analysis of recognition rates for word and subword-based speech recognition, see for example [7].

There are many possible ways to select a subword vocabulary. Morphologically motivated units like statistical morphs have proven to be a good choice. By definition, morphemes are the smallest meaning-bearing units of a language and morphs are their realizations in text or speech. Different algorithms for morphological segmentations have been evaluated in Morpho Challenge competitions [8] that have included speech recognition tasks [3]. For comparing the prediction ability of different subword vocabularies, a useful measure is the cross-entropy of an n-gram model trained for the subword units. In this work, we study whether the unigram likelihood would be a suitable criterion for learning a subword vocabulary that would produce an accurate high-order n-gram model for an LVCSR task. We introduce an unsupervised segmentation method that explicitly optimizes unigram likelihood for given vocabulary size. We also compare the unigram entropy and morphological correctness of the subwords as the criteria for achieving a high-order model with low entropy.

2. GREEDY ALGORITHM FOR SUBWORD VOCABULARY LEARNING

The goal of the suggested new algorithm is to create a subword vocabulary that gives a high unigram likelihood for the training corpus. This criterion is difficult to optimize directly for a limited vocabulary size, but the Greedy 1-Grams ($G1G$) algorithm that we present, provides a reasonable approximation. The approach taken is to start with a large amount of

This work was financially supported by the Academy of Finland under the Finnish Centre of Excellence Program 2012–2017 (grant no. 251170) and LASTU Programme (nos. 256887 and 259934). We thank Dr. Teemu Hirsimäki, Dr. Vesa Siivola, and Dr. Mathias Creutz for fruitful discussions and their help in developing the algorithm.

candidate subwords and gradually constrain the subword vocabulary and refine word segmentations to reject subwords that are least significant for the likelihood.

The algorithm takes as input a list of words and their frequencies. The subword vocabulary V consists of subwords and a probability for each subword, forming a unigram distribution over the vocabulary. The model structure is explained in more detail in Subsection 2.1. The algorithm consists of separate initialization and pruning phases which are explained in more detail in Subsections 2.2 and 2.3.

2.1. Markov models and subword segmentation

Generating words from a subword unigram model may be viewed as a zero-order Markov process. However, with respect to inferring model parameters from a set of words, the states of the process are not directly observable, because the states emit subwords of varying length and the borders between the subwords are not observed. Extending the Viterbi and Forward-Backward algorithms for this multigram framework [9] is straightforward. Figure 1 shows an example how Finnish word "talossa" could be segmented as a sequence of letters, subwords, or as a single observation. The most likely segmentation returned by Viterbi would in this case be "talo + ssa".

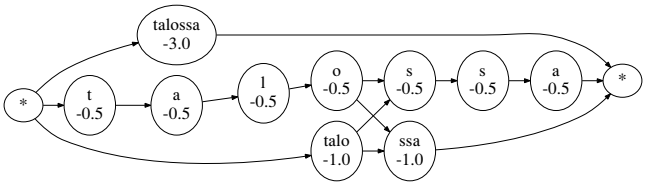

Fig. 1. Segmentation paths for word "talossa" using a subword model. The numbers are log-likelihoods of the units.

The vocabulary may be stored in a letter-trie-like data structure to allow fast subword lookups starting from each character position.

2.2. Initialization

The algorithm proceeds by starting with a large vocabulary, which is then pruned to a suitable size. Because no new subwords will be introduced, proper initialization is important.

1. Train a letter n-gram model from the training corpus. Forward-backward algorithm in step 4 decreases the differences between different discounting methods.

2. Select the initial pool of subwords $V = \{s_i\}$, for example all substrings from the most common words in the training data up to a reasonable maximum length.

3. Calculate a log-probability lp_i for each subword s_i using the letter n-gram model. For example, if the subword s_i is a four-letter string "abcd":

$$lp_i(abcd) = lp(a) + lp(b|a) + lp(c|ab) + lp(d|abc) \quad (1)$$

   Normalize the probabilities to sum to one.

4. Using the subword probabilities, iterate Forward-backward over the training corpus until convergence. In practice, around 5 iterations is enough. Update the probabilities for all subwords.

5. Iterate Viterbi training. After each iteration increase cutoff value and remove subwords with frequency below the cutoff value. Stop when a suitable maximum cutoff value is reached.

2.3. Vocabulary pruning

The actual pruning is done with a more refined pruning strategy, which tries to account for the effect that removing a subword has on the likelihood. For segmentation, either Forward-backward or Viterbi algorithm may be used. We did not observe significant difference between them, and thus Viterbi segmentation was used. Iterate:

1. Resegment all words, update subword probabilities and store pointers from subwords to words.

2. Select a list of candidate subwords for removal, for example the least frequent subwords in the vocabulary.

3. For each candidate subword, estimate the cost of removing it by resegmenting the words without it.

4. Sort the list of candidate subwords in descending order by the value of estimated likelihood change.

5. Remove a defined amount of top candidate subwords. Alternatively it is possible to update parameters after each removal and verify that the cost for each subsequent removal is above a threshold value.

3. EXPERIMENTS

3.1. Experimental setup

The text corpus used in all further experiments is the Finnish CSC Kielipankki corpus [10]. It contains text from Finnish newspapers, magazines and books. The corpus contains in total 144 million word tokens and 4.1 million word types. The corpus was preprocessed by removing punctuation and other special characters and numbers were expanded to their written form. The order of the sentences was randomized and the sentences divided into training, development and evaluation sets. Training set consists of 139 million word tokens. It is

used for training the subword vocabularies and n-gram models. The development set consists of 190000 words tokens and meant for optimizing discount parameters of n-gram models. The evaluation set consists of 3.9 million word tokens and is used in evaluating the entropy of the n-gram models.

The speech recognition task is based on the Finnish Speechdat database[1], which consists of 4000 speakers recorded over fixed telephone line (8 khz). 55 hours from 3696 speakers were used for training. For both development and test sets 150 separate speakers were allocated with about 2.2 hours of speech in each set. The task is relatively hard because of wide range of speakers and at times low sound quality.

The speech recognition system is a large vocabulary speech recognizer, which utilizes Hidden Markov Models for acoustic modeling and n-grams for language modeling. The HMM is based on context-dependent triphones with 1783 tied HMM states. State emission PDFs are diagonal Gaussian mixture models with varying numbers of Gaussian trained using maximum likelihood training with a global MLLR linear transform. The total number of Gaussian was 85758. No speaker adaptation was utilized in these experiments. To be able to use subword units and allow long-context n-grams, there are special issues to consider in the decoder implementation [11]. Letter-to-phoneme mapping for Finnish is quite straightforward and using triphone models solves most of the issues.

3.2. Subword vocabularies

3.2.1. G1G

The initial pool of subwords was selected by taking all substrings up to length of 15 from the $500k$ most common words in the training data. The initial number of subwords was 4.8 million. A letter 8-gram model was trained using the SRILM toolkit [12]. Each of the strings was then assigned an initial unigram score using the letter n-gram model. From this point on $1M$ most common words were used as the training data, as the remainder had negligible effect on the result. Forward-backward algorithm was iterated 5 times. After this we changed to Viterbi segmentation, removing all unused subwords. The vocabulary size dropped to $493k$ subwords. Viterbi segmentation was then iterated 20 times while increasing a cutoff value in steps of 2.5 to a maximum of 50. This further reduced the vocabulary size to $177k$ subwords.

Pruning was done as follows: in each iteration, a list of candidate removals of size 25k was created and sorted using the estimate of their impact on the likelihood. Before reaching vocabulary size of $100k$, 2500 subwords were removed per iteration. For vocabulary sizes 50–$100k$, 1000 subwords were removed per iteration and for vocabulary sizes 10–$50k$, 500 subwords were removed per iteration.

[1] http://www.speechdat.fi

Table 1 illustrates the development of subword vocabulary size and unigram log-likelihood after subsequent steps in $G1G$ training. The last steps show the points in iteration where the size reached $100k$, $50k$, $25k$ and $10k$.

Table 1. G1G training statistics (LL = log-likelihood).

Step	**Vocabulary size**	**LL** (10^9)
Initialization	4.8M	-1.58
Forward-backward (5 iter)	4.8M	-1.39
First Viterbi segmentation	493k	-1.39
Cutoff 50	177k	-1.42
Iteration 32	100k	-1.45
Iteration 81	50k	-1.51
Iteration 131	25k	-1.58
Iteration 161	10k	-1.72

3.2.2. Morfessor

Morfessor vocabularies were created with the Morfessor Baseline script [13]. Unweighted lists of common words were given as input to the algorithm. Longer word list results in a larger vocabulary and vice versa. To obtain the results of this paper, we took $50k$-–$600k$ most common words in the training corpus.

For longer word lists with word frequencies, Morfessor algorithm segments very few words. The underlying reason for the undersegmentation is that Morfessor is based on the MDL two-part criterion, having a cost for the vocabulary and the data. The cost for the vocabulary must be on a meaningful level compared to the data cost. It is possible to guide the algorithm to segment more by setting a larger weight for the lexicon cost. Lexicon-weighted Morfessor has been evaluated with respect to morphological F-measure in [14], but no prior evaluation for language modeling and LVCSR exists.

The lexicon-weighted Morfessor segmentations were created using a frequency weighted list of $500k$ most common words in the training corpus. Word list of $1M$ words was noticed to result in a slight performance degradation. The lexicon size was controlled by varying the weighting term between corpus and lexicon codelengths. In these experiments the lexicon weight was varied between 50–400.

3.3. F-measure for morphological segmentation

The methods were evaluated in terms of morphological F-measure using the BPR evaluation and gold standard segmentations described in [15]. F-measure is defined as the harmonic mean of precision (P) and recall (R) for the placement of morph boundaries:

$$P = H/(H + I); \qquad R = H/(H + D), \qquad (2)$$

where H is the number of correct boundary positions, I is the number of incorrect boundary positions, and D is the number of missing boundary positions when compared to the gold standard segmentation.

The results for vocabulary sizes of $35k$ are shown in Table 2. As the vocabulary size has been selected for speech recognition task, rather than for morphological evaluation, these are not the best possible F-measures for the methods. While Morfessor Baseline trained without word frequencies reached the best F-measure, also Morfessor LW and $G1G$ were reasonably accurate.

Table 2. Morphological F-measures for the methods.

Method	Precision	Recall	F-Measure
Morfessor	0.73	0.52	0.61
Morfessor LW	0.69	0.47	0.56
G1G	0.70	0.48	0.57

3.4. Entropy evaluation

Language models were trained using the VariKN toolkit [16]. Models are Kneser-Ney smoothed and trained using a growing algorithm. The training data was processed to contain a special word boundary symbol between all words and also in the beginning and end of sentences. An example training sentence would thus look like:

```
<s> <w> kissa <w> käve li <w> kadu lla <w> </s>
```

Average word entropy is a good measure of how well the language model can predict the words and word sequences in the evaluation set [17]. Entropy was evaluated for each n-gram model as follows:

1. Segment the evaluation corpus $\mathcal{C}$ with the n-gram model $\mathcal{M}$ into subwords in similar format as the training data. This was done using a segmenter, which finds the most likely subword segmentation for each sentence, given the model $\mathcal{M}$.

2. Average word entropy for the evaluation corpus was then computed with the following formula:

$$H_{\mathcal{M}}(\mathcal{C}) = \frac{1}{|C|}\sum_{s\in C} -\frac{1}{W_s}\sum_{i\in[2,|T_s|]} \log_2 P(t_i|t_0^{i-1},\mathcal{M}), \tag{3}$$

where $|C|$ is the total number of sentences, W_s number of words in the sentence s and T_s all text tokens in the sentence s. The cost for the two first tokens is omitted as they are the same for each sentence. Word normalization is important as token-wise entropy is not meaningful when comparing different vocabularies. The result is in bits per word.

In the first experiment, unigram entropy of the evaluation corpus was evaluated as a function of the vocabulary size. The results are in Figure 2. It can be seen that there are relatively large differences between the methods. It seems that $G1G$ performs best and the Morfessor Baseline worst, lexicon-weighted Morfessor reaching roughly the same level as $G1G$. Comparing the morphological F-measures in Table 2 and the unigram entropies in Figure 2, it may be noticed that for the evaluated methods, unigram entropy and morphological F-measure are negatively correlated.

In the second experiment, the vocabulary size was set to roughly the same number for all methods, and variable-length high order n-gram models were trained with the VariKN toolkit. The maximum order of n-grams was set to 10 for all models. N-gram frequency cutoff 2 was used for all models and all n-gram orders ranging from 2–10. In preliminary tests, this gave the best models. Model sizes were controlled by varying the pruning parameters. Word entropy was measured as a function of the n-gram model size. The results are in Figure 3. $G1G$ and Morfessor LW perform better than baseline Morfessor method for most of the model sizes. The entropy curves were found to be quite invariant to even large changes in vocabulary size.

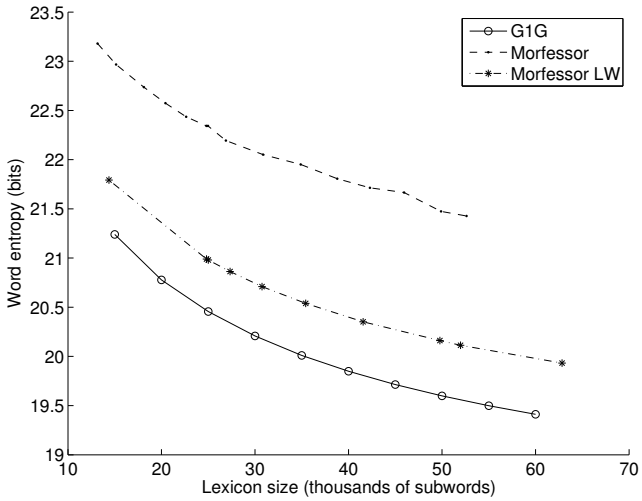

Fig. 2. Word unigram entropy as a function of vocabulary size.

3.5. Finnish LVCSR task

The language models in this task are the same as in the entropy evaluation. One model with around 16 million n-grams was chosen for each method. In addition, 2-gram lookahead models were trained for each method. The development set was used to optimize the language model scale. Real-time factor for decoding was evaluated on an Intel Xeon E3-1230 3.30GHz CPU using precomputed acoustic probabilities.

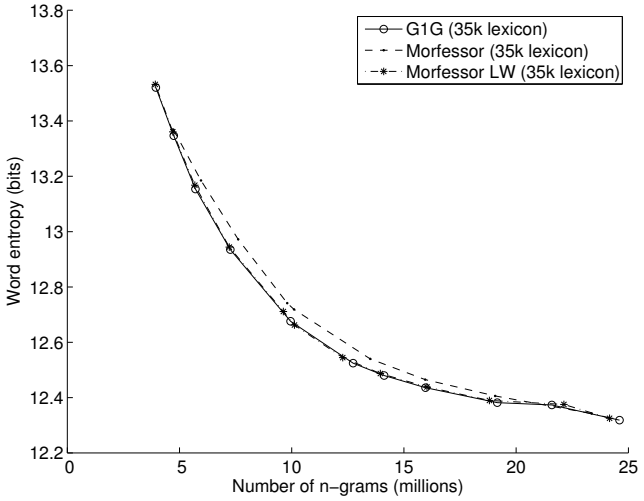

Fig. 3. Word varigram (max. 10g) entropy as a function of number of n-grams in the model.

Table 3. Results in a Finnish LVCSR dictation task

Method	Vocab	n-grams	LER	WER	RTF
Morfessor	35k	16.0 M	7.40	22.32	2.41
Morfessor LW	35k	16.0 M	7.25	21.88	2.34
G1G	35k	16.0 M	7.18	22.02	2.15

The results can be found in Table 3. Because the Finnish words are quite long and typically contain several morphemes, the most widely used performance measure is letter error rate (LER). Prior to this work, Morfessor Baseline has been the best performing method in our setup. Training Morfessor with frequencies and by weighting lexicon, a decrease in both LER and real time factor (RTF) was recorded. $G1G$ algorithm further improves both metrics. We experimented also with model sizes of around 10 and 24 million n-grams and the conclusions were similar. The improvements in error rates are modest, but quite clear because of the large testset. Also, around 10% improvement in decoding speed signifies that the model is consistently suggesting better hypotheses.

4. DISCUSSION

The purpose was to test whether a good subword vocabulary could be constructed by minimizing the average word entropy for a given subword vocabulary size. As directly optimizing the high order n-grams required by LVCSR is computationally hard, we approximated it by trying to minimize the unigram entropy of the training corpus.

High morphological accuracy is a good property for a subword vocabulary for many natural language processing tasks [15]. Unigram entropy and F-measure are probably somewhat dependent, as all the methods tried in this work performed reasonably well in both respects. Our results suggest that unigram entropy is more important than morphological correctness if the goal is to train a high-order n-gram model for a speech recognition task.

The proposed approach of starting with a large pool of candidate substrings and pruning it to a suitable size provided the most efficient subword vocabulary for a Finnish LVCSR task. Our experiments also showed that training Morfessor with word frequencies by weighting the lexicon cost improved over the baseline Morfessor. A possible future work is system combination with $G1G$ and Morfessor based subword models, because both methods give good recognition results, but with different subwords and recognition errors.

As the proposed $G1G$ algorithm is quite general, it should be easy to extend in various ways:

Scalability to longer strings. As the number of allowed substrings is limited, it is possible to scale the algorithm for longer strings. This is more difficult with approaches such as Morfessor, that introduce new strings during training. In the context of language modeling, a possible generalization would be segmenting sentences into more general chunks of text, allowing cross-word segments and multiwords (cf. [9, 18]). In many languages, through declension, the suffix of previous word is connected to the adjacent word. These cases might be better modelled by allowing the units to cross word boundaries. At least in conversational LVCSR, it is a relatively common practice to use multiword units [19]. $G1G$ algorithm could be used to learn the most important multiwords on text-level. With suitable modifications, the algorithm could be used in segmenting continuous strings, not limited to natural language.

Pronunciation variants. By removing subwords instead of introducing new ones, it should be easy to control the quality of pronunciation lexicon for languages with more complex letter-to-phoneme mapping. The vocabulary could be initialized by selecting only strings with a well defined or estimable pronunciation variant and record the possible changes on word-level while pruning the vocabulary.

Higher-order statistics. If the goal is to optimize the entropy of a high-order n-gram model, utilizing higher-order statistics [20] already in the vocabulary training phase could improve results. Bigram statistics have been tried for instance in Chinese word segmentation [21]. In our initial experiments for subword segmentation, starting with a large vocabulary seemed to avoid local maxima rather well.

5. CONCLUSIONS

Our results suggest that unigram entropy is a good indicator for the quality of high order n-gram models for that vocabulary. A novel algorithm, $G1G$, which learns a subword vocabulary based on unigram likelihood, was presented. It provided the best performing subword vocabulary for a Finnish LVCSR task. The algorithm is quite general by nature and could prove useful for other string segmentation and compression tasks.

6. REFERENCES

[1] R. Sarikaya, K. Kirchhoff, T. Schultz, and D. Hakkani-Tür, "Introduction to the special issue on processing morphologically rich languages," *IEEE Transactions On Audio, Speech, and Language Processing*, vol. 17, no. 5, pp. 861–862, 2009.

[2] T. Hirsimäki, J. Pylkkönen, and M. Kurimo, "Importance of high-order N-gram models in morph-based speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 17, no. 4, pp. 724–732, 2009.

[3] M. Kurimo, M. Creutz, M. Varjokallio, E. Arisoy, and M. Saraçlar, "Unsupervised segmentation of words into morphemes - Morpho Challenge 2005: Application to automatic speech recognition," in *INTERSPEECH*, Pittsburgh, USA, 2006.

[4] P. Mihajlik, Z. Tüske, B. Tarján, B. Németh, and T. Fegyó, "Improved recognition of spontaneous Hungarian speech — morphological and acoustic modeling techniques for a less resourced task," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1588–1600, Aug 2010.

[5] M. Creutz and K. Lagus, "Unsupervised discovery of morphemes," in *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, Philadelphia, Pennsylvania, USA, 2002, pp. 21–30.

[6] M. Creutz, A. Stolcke, T. Hirsimäki, M. Kurimo, A. Puurula, J. Pylkkönen, V. Siivola, M. Varjokallio, E. Arisoy, and M. Saraçlar, "Morph-based speech recognition and modeling of out-of-vocabulary words across languages," *ACM Transactions on Speech and Language Processing*, vol. 5, no. 1, pp. 1–29, 2007.

[7] T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, and J. Pylkkönen, "Unlimited vocabulary speech recognition with morph language models applied to finnish," *Computer Speech and Language*, vol. 20, no. 4, pp. 515–541, 2006.

[8] M. Kurimo, S. Virpioja, V. Turunen, and K. Lagus, "Morpho Challenge 2005-2010: Evaluations and results," *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pp. 87–95, 2010.

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

[10] University of Helsinki; University of Joensuu; CSC Scientific Computing Ltd, "Kielipankki corpus. An electronic document collection of the Finnish language," .

[11] J. Pylkkönen, "An efficient one-pass decoder for Finnish large vocabulary continuous speech recognition," in *Proceedings of the 2nd Baltic Conference on Human Language Technologies*, 2005.

[12] A. Stolcke, J. Zheng, W. Wang, and V. Abrash, "SRILM at sixteen: Update and outlook," in *IEEE Automatic Speech Recognition and Understanding Workshop*, 2011, p. 5.

[13] M. Creutz and K. Lagus, "Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0," Tech. Rep. A81, Publications in Computer and Information Science, Helsinki University of Technology., 2005.

[14] S. Virpioja, O. Kohonen, and K. Lagus, "Evaluating the effect of word frequencies in a probabilistic generative model of morphology," in *Proceedings of NODALIDA 2011, volume 11 of NEALT Proceedings Series*, Riga, Latvia, 2011, pp. 230–237.

[15] S. Virpioja, V. T. Turunen, S. Spiegler, O. Kohonen, and M. Kurimo, "Empirical comparison of evaluation methods for unsupervised learning of morphology," *Traitement Automatique des Langues*, vol. 52, no. 2, pp. 45–90, 2011.

[16] V. Siivola, T. Hirsimäki, and S. Virpioja, "On growing and pruning Kneser-Ney smoothed N-gram models," *IEEE Transactions on Speech, Audio and Language Processing*, vol. 15, no. 5, pp. 1617–1624, 2007.

[17] J. T. Goodman, "A bit of progress in language modeling. Extended Version.," Tech. Rep., Microsoft Research, 2001.

[18] D. Klakow, "Language-model optimization by mapping of corpora," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Seattle, WA, USA, may 1998, vol. 2, pp. 701–704, IEEE.

[19] A. Stolcke et al., "The SRI March 2000 Hub-5 conversational speech transcription system," in *NIST Speech Transcription Workshop*, 2000.

[20] S. Deligne and Y. Sagisaka, "Learning a syntagmatic and paradigmatic structure from language data with a bi-multigram model," in *COLING-ACL*, 1998, pp. 300–306.

[21] S. Goldwater, T. L. Griffiths, and M. Johnson, "Contextual dependencies in unsupervised word segmentation," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, 2006.