Conversion of Recurrent Neural Network Language Models to Weighted Finite State Transducers for Automatic Speech Recognition

Gwénolé Lecorvé, Petr Motlicek

Idiap Research Institute, Martigny, Switzerland
gwenole.lecorve@idiap.ch, petr.motlicek@idiap.ch

Abstract

Recurrent neural network language models (RNNLMs) have recently shown to outperform the venerable n-gram language models (LMs). However, in automatic speech recognition (ASR), RNNLMs were not yet used to directly decode a speech signal. Instead, RNNLMs are rather applied to rescore N-best lists generated from word lattices. To use RNNLMs in earlier stages of the speech recognition, our work proposes to transform RNNLMs into weighted finite state transducers approximating their underlying probability distribution. While the main idea consists in discretizing continuous representations of word histories, we present a first implementation of the approach using clustering techniques and entropy-based pruning. Achieved experimental results on LM perplexity and on ASR word error rates are encouraging since the performance of the discretized RNNLMs is comparable to the one of n-gram LMs.

Index Terms: Language model, recurrent neural network, weighted finite state transducer, speech decoding

1. Introduction

Recurrent neural network language models (RNNLMs) have shown to outperform the venerable n-gram language models (LMs) [1]. However, in automatic speech recognition (ASR), RNNLMs cannot be used to directly decode a speech signal since they rely on continuous representations of word histories while decoding algorithms require to handle discrete representations to remain tractable [2, 3]. Instead, RNNLMs are currently used to rescore N-best lists generated using n-gram LMs. Hence, the prediction power of RNNLMs is used only on subsets of all transcription hypotheses. Such an approach does not offer the optimal solution since the n-gram LM used for the decoding may have discarded hypotheses which the RNNLM would have judged very likely. Furthermore, the distributions of these two kinds of LMs have been shown to be complementary [4, 5]. The use of RNNLMs in early stages of speech decoding is thus a challenging objective.

Recently, few studies were devoted to this problem. In [6], the authors propose to sample word sequences by using RNNLM as a generative model before training an n-gram LM based on the generated text. By exploiting this LM to perform first pass decoding, achieved results outperformed the use of n-gram LMs trained on standard texts. However, we assume that this approach is still not optimal since it still prevents from relying on long span information during the decoding. In [7], the author has proposed an *iterative decoding* algorithm which enables to efficiently rescore word lattices using RNNLMs. The main idea is to partition word lattices to reduce the computational complexity of browsing all possible hypotheses. Though leading to good results, this technique cannot be directly applied in the first pass of the decoding since no explicit search graph is available at this moment of the recognition process.

In this paper, we define a new generic strategy to transform RNNLMs into a Weighted Finite State Transducer (WFST) which can directly be used within the decoding process of an ASR system [3]. We believe that this approach has a potential to outperform the conventional approach where RNNLMs are employed to rescore N-best hypotheses as a final step of ASR. The principle of the conversion consists in discretizing continuous RNNLM representations of word histories in order to build WFST states, and then to link these states with probabilities derived from the RNNLM. In practice, this approach also raises some needs for pruning the generated WFST since the theoretical number of states may be large according to the chosen discretization strategy. We present a preliminary implementation of the RNNLM conversion algorithm based on K-means clustering and entropy pruning.

This paper is organized as follows: after recalling the principles of RNNLMs provided in Section 2, the generic conversion strategy is introduced in Section 3. Section 4 presents how K-means clustering and entropy pruning can be used to implement a first version of the generic strategy. Finally, Section 5 describes experiments on the Penn Treebank corpus and using LVCSR meeting system.

2. Overview of RNNLMs

The goal of a language model employed in ASR system is to provide the conditional probability of a word w_i given an history h of preceding words. As detailed in [1], this history is represented in RNNLMs by the most recent preceding word w_{i-1} and a multi-dimensional continuous representation $\mathbf{c}_i$ of the remaining context. The topology of the neural network used to compute conditional probabilities $P[w_i|w_{i-1}, \mathbf{c}_{i-1}]$ is organized in 3 layers using a bottleneck scheme. The input layer reads a word w_{i-1} and a continuous history $\mathbf{c}_{i-1}$. The hidden layer compresses the information of these two inputs and computes a new representation $\mathbf{c}_i$. The value $\mathbf{c}_i$ is then passed to the output layer which, after normalization, provides the conditional probabilities.

3. Generic RNNLM conversion

The goal of this work is to convert RNNLMs into an approximate WFST representation. As illustrated in Figure 1, this task mainly consists in binding discrete states with the continuous input states of the RNNLM, and in using these discrete states as the nodes of a WFST. The edges among nodes are then labeled with word transitions and their probabilities estimated by the RNNLM. There are two key aspects to achieve this task. First, a *discretization function* needs to be defined to transform the continuous representations. Second, we have to take into account the size of the output WFST since enumerating all possible discrete states might quickly be untractable as soon as the vocabulary becomes large and the discretization becomes precise. Thus, a *pruning criterion* needs to be defined in order to discard uninteresting discrete states, and a *back-off strategy* in order to model the pruned states in a simpler way. This sec-

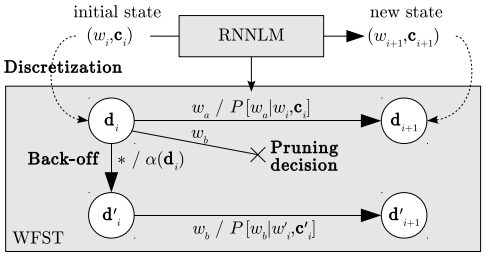

Figure 1: *Overview of the RNNLM discretization scheme.*

tion formally introduces these parameters before providing the generic algorithm for the conversion.

3.1. Discretization function

The main function to be defined is a discretization function which returns a discrete state for every possible input continuous state $(w_i, \mathbf{c}_i)$. The generic form of this discretization function f is as follows:

$$\begin{array}{rccccc} f: & V & \times & \mathbb{R}^h & \longrightarrow & \mathbb{N}^k \\ & w_i & , & \mathbf{c}_i & \longmapsto & \mathbf{d}_i \ , \end{array} \tag{1}$$

where V is the LM vocabulary, h is the size of the hidden layer of the RNNLM, and k is the dimension of the discrete representation $\mathbf{d}_i$. As it will be shown in the algorithm, we also need to be able to go back from discrete states to continuous representations. Thus, the partial inverse function f^{-1} must also be defined such that:

$$\begin{array}{rccc} f^{-1}: & \mathbb{N}^k & \longrightarrow & Y \subset V \times \mathbb{R}^h \\ & \mathbf{d}_i & \longmapsto & (w_i, \mathbf{c}_i) \ . \end{array} \tag{2}$$

This function is only a partial inverse of function f since f is, by definition, a surjection. Thus, the codomain of f^{-1} is limited to a subset of $V \times \mathbb{R}^h$.

3.2. Pruning criterion and back-off

Since the WFST is intended to be used in ASR during the decoding, it should not be too large. Thus, it is important to be able to control the size of the WFST by pruning and by introducing a back-off scheme. Furthermore, since the cardinality of the discrete domain $\mathbb{N}^k$ can be huge, the pruning should be done on the fly, i.e., while building the WFST. This requires to define two parameters.

First, a pruning criterion π must be defined to decide whether a corresponding edge should be discarded or added when building the WFST. A "good" pruning criterion should be such that the pruning of an edge does not lead to large information loss. Given a node $\mathbf{d}_i$, the criterion should thus be derived from the quantity of information carried by a new word transition v. The generic form of this criterion is

$$\pi(v, \mathbf{d}_i) = \begin{cases} \textit{false} & \text{if } P(v|f^{-1}(\mathbf{d}_i)) \text{ is informative enough,} \\ \textit{true} & \text{otherwise .} \end{cases} \tag{3}$$

Second, a back-off mechanism must be introduced in order to approximate the probability of pruned events[1]. Basically, this strategy requires to define a back-off function β which maps a given discrete representation to a simpler representation:

$$\begin{array}{rccc} \beta: & \mathbb{N}^k & \longrightarrow & \mathbb{N}^k \\ & \mathbf{d}_i & \longmapsto & \mathbf{d}'_i \ . \end{array} \tag{4}$$

[1] Even if an event is judged as unlikely, it does not mean that it cannot occur. Hence, the model must be able to provide a probability for any event.

Data: L, a list of discrete states, i.e., of WFST nodes
$L \leftarrow f(\text{beginning of sentence})$;
while $L \neq \emptyset$ **do**
  $\mathbf{d}_{\text{src}} \leftarrow \text{pop}(L)$;
  $(w_{\text{src}}, \mathbf{c}_{\text{src}}) \leftarrow f^{-1}(\mathbf{d}_{\text{src}})$;
  $\mathbf{c}_{\text{dst}} \leftarrow \text{hidden\_layer}(w_{\text{src}}, \mathbf{c}_{\text{src}})$;
  foreach $v \in V$ **do**
    if $\mathbf{d}_{src} = \mathbf{d}_{min}$
    or not $\pi(v, \mathbf{d}_{\text{src}})$ **then**
      $p \leftarrow P(v|w_{\text{src}}, \mathbf{c}_{\text{src}})$;
      $\mathbf{d}_{\text{dst}} \leftarrow f(v, \mathbf{c}_{\text{dst}})$;
    else
      $p \leftarrow 0$;
      $v \leftarrow \epsilon$;
      $\mathbf{d}_{\text{dst}} \leftarrow \beta(\mathbf{d}_{\text{src}})$;
    if node $\mathbf{d}_{\text{dst}}$ does not exist **then**
      add_node_to_wfst($\mathbf{d}_{\text{dst}}$);
      push($L, \mathbf{d}_{\text{dst}}$);
    add_edge_to_wfst($\mathbf{d}_{\text{src}} \xrightarrow{v:v,p} \mathbf{d}_{\text{dst}}$);
compute_backoff_weights();

Algorithm 1: *Pseudo-code of the RNNLM conversion.*

The destination state $\mathbf{d}'_i$ is referred to as the *back-off state* or *node* of the state $\mathbf{d}_i$. To avoid cycles in the WFST, the function β must define a partial order over all discrete states, i.e., it must guarantee that $\mathbf{d}'_i$ is "simpler" than $\mathbf{d}_i$. This naturally introduces the notion of minimal state $\mathbf{d}_{\text{min}}$, i.e., the state for which no pruning and back-off can be done.

3.3. Conversion algorithm

Assuming that the discretization function f, the pruning criterion π, and the back-off function β are defined, the conversion algorithm seeks to discretize each given RNNLM state and to build outgoing edges reaching new states. This process can be written in an iterative way whose pseudo-code is given by the Algorithm 1. Given the list of states which have already been added to the WFST but for which no outgoing edge has been created yet, the algorithm pops a state, computes probabilities using the RNNLM, and then builds edges. As soon as an edge reaches a new destination node in the WFST, this next node is built and pushed into the list on remaining states to be explored. In practice, the conversion process starts with the RNNLM state corresponding to a beginning of sentence. When considering pruning, a decision must be made before adding a new edge. If the edge starts from the minimal state or carries enough information, then it is built. Otherwise, it is discarded and redirected to a back-off node. The weights of back-off transitions are computed after building the WFST by following Katz's strategy [8].

4. Implementation

We propose to implement the generic process described above by using K-means clustering for the discretization of RNNLM states and entropy-based criteria for the pruning strategy. This implementation is described in this section.

4.1. Discretization using K-means

We propose to discretize RNNLM states by first partitioning their continuous domain into clusters computed using the K-means algorithm, and then by associating each state to a corresponding cluster. Each cluster is denoted by an identifier and is associated with its centroid. Given a set of K centroids, the

discretization and "undiscretization" functions are defined as:

$$f_K(w\ ,\ \mathbf{c}) = (\ w\ ,\ k\ )\ , \tag{5}$$

where $k \in [\![1, K]\!]$ is the ID of the centroid associated with $\mathbf{c}$, and

$$f_K^{-1}(w\ ,\ k) = (\ w\ ,\ \mathbf{c}_k\ )\ , \tag{6}$$

where $\mathbf{c}_k$ is the k-th centroid. As mentioned in Section 3.1, we can clearly see that, in most cases, $f_K^{-1}\big(f_K(x)\big)$ does not equal to x, which means that some information is lost.

An advantage of K-means is that the size of the discrete space of the WFST nodes can be explicitly set through K. Nonetheless, for a large vocabulary, the size of the final WFST can be huge if no pruning is applied[2]. To train the centroids, the RNNLM is first applied on a text data, e.g., the training text. Then, each continuous state $\mathbf{c}_i$ encountered is stored and the K-means clustering is performed on these logs.

4.2. Back-off

We define a two-fold back-off scheme such that the information about the long-span context is dropped as first and the information about the last word is dropped as second. Formally, given a discrete state (w, k), this consists in defining $\beta(w, k) = (w, \varnothing)$ and $\beta(w, \varnothing) = (\varnothing, \varnothing)$ where $\varnothing$ means that no information is provided. To remain compliant with the method, values are defined according to f^{-1} for these two special discrete states: $f^{-1}(w, \varnothing) = (w, \mathbf{c}_0)$ where $\mathbf{c}_0$ is the global mean of all the continuous states observed during the K-means clustering, and $f^{-1}(\varnothing, \varnothing) = (\varnothing, \mathbf{c}_0)$.

4.3. Pruning

Within the conversion process, the principle of pruning is to reduce the final WFST size by not building edges whose absence does not result in significant information loss. A well known strategy for this problem consists in identifying edges which have a minimal effect on the entropy of the probability distribution [9]. Following this principle, we define our pruning criterion based on two values.

First, the piece of entropy carried by a transition from the state $\mathbf{d}$ with the word v is considered. It is expressed as:

$$H(v, \mathbf{d}) = -P(v, f_K^{-1}(\mathbf{d})) \cdot \log P(v, f_K^{-1}(\mathbf{d}))\ . \tag{7}$$

By denoting $P(v|f_K^{-1}(\mathbf{d}))$ to $P(v|\mathbf{d})$, the joint probability $P(v, f_K^{-1}(\mathbf{d}))$ of a word v and its history $\mathbf{d}$ can be approximated as:

$$P(v, f_K^{-1}(\mathbf{d})) \approx P(v|\mathbf{d}) \cdot P(\mathbf{d}) \tag{8}$$

$$= P(v|w, \mathbf{c}_k) \cdot P(w, k)\ . \tag{9}$$

The probability $P(v|w, \mathbf{c}_k)$ is directly given by the RNNLM while the probability $P(w, k)$ is considered as a prior estimated from the logs used to train the centroids. Since the estimation of this joint probability may be unreliable because of data sparsity, an independence assumption between w and k is made. In practice, $P(w, k)$ is thus simplified to $P(w) \cdot P(k)$.

Second, an important aspect is to know if the probabilities of an event remain close before and after pruning. For a transition $(v, \mathbf{d})$, the relative difference between these two probabilities is defined as:

$$D(v, \mathbf{d}) = \frac{|P(v|\mathbf{d}) - \alpha(\mathbf{d}) \cdot P(v|\beta(\mathbf{d}))|}{P(v|\mathbf{d})}\ , \tag{10}$$

[2]Precisely, the theoretical maximum numbers are $|V| \times K$ nodes and $(|V| \times K)^2$ edges.

Table 1: *Perplexities of n-gram LMs and of the RNNLM.*

2-grams	3-grams	4-grams	5+-grams	RNNLM
186	148	142	141	124

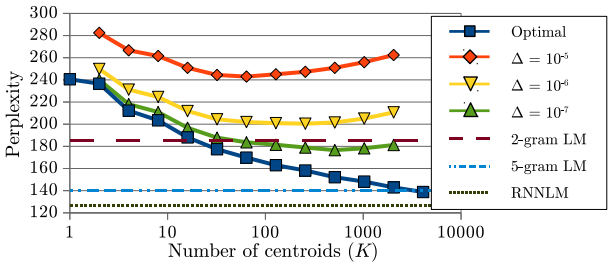

Figure 2: *Perplexities on test set of the Penn Treebank corpus for WFSTs generated using various numbers of centroids and various pruning thresholds Δ.*

where $\beta(\mathbf{d})$ is the back-off state for $\mathbf{d}$, and the back-off weight $\alpha(\mathbf{d})$ is approximated by iteratively estimating the probability mass of events which will be pruned for the state $\mathbf{d}$. The higher $D(v, \mathbf{d})$, the lesser backing off preserves the original probability.

Finally, for a node $\mathbf{d}$ and a transition word v taken under examination, we define the pruning criterion as follows:

$$\pi(w, \mathbf{d}) = \begin{cases} \textit{false} & \text{if } H(v, \mathbf{d}) \cdot D(v, \mathbf{d}) < \Delta \\ \textit{true} & \text{otherwise}\ , \end{cases} \tag{11}$$

where Δ is a user-determined pruning threshold (the lower the value, the larger the size of the WFST).

The whole process has been implemented using the RNNLM toolkit[3] and the OpenFst library[4]. Conversions last between a few minutes and a few hours according $|V|$, K, and Δ.

5. Experiments

Two series of experiments have been carried out to evaluate the proposed approach: (a) experiments on the Penn Treebank corpus to study the behavior of the conversion process, and (b) the decoding experiments on meeting speech recordings using a large vocabulary continuous speech recognition system.

5.1. Perplexities on the Penn Treebank corpus

The goal of the first part of experiments is to study an impact of the K-means algorithm as well as the pruning threshold on the RNNLM conversion. To do so, we use the same LMs as those used in [1] on the Penn Treebank corpus. Two types of LMs are considered: n-gram LMs trained with various orders using maximum likelihood estimation and Kneser-Ney smoothing, and a RNNLM based on a hidden layer of 300 neurons. The Penn Treebank corpus is a portion of the Wall Street Journal which is widely used for evaluating performance of statistical LMs [10]. This corpus is split into 3 parts: a training set of 900K words, a development set of 70K words (which is only used for RNNLM training), and a test set of 80K words. The vocabulary is made of 10K words. Perplexities of these models on the test set are reported in Table 1.

Various values of K have been used to extract centroids from the training set, as described in Section 4.1. Furthermore, 3 different values have been set for the pruning threshold. WFSTs are generated using these settings and the final perplexities are measured on the test set. These perplexities are reported in Figure 2 and are compared with those of the other models.

[3]http://www.fit.vutbr.cz/~imikolov/rnnlm/

[4]http://www.openfst.org

Table 2: *Perplexities of LMs on the evaluation set of RT 2007.*

2-gram LM	4-gram LM	WFST	RNNLM
162	93	127	93

Additionally, the optimal perplexity, which can be obtained if no pruning was applied, is given in Figure 2. First, it appears that this optimal value decreases when the number K of centroids increases (as increasing K means that richer history can be considered). Although the optimal perplexity does not reach the perplexity of the original RNNLM, these preliminary results interestingly show that the discretization does not lead to large information loss as long as K is large enough. Then, a degradation can clearly be observed when introducing pruning (i.e., $\Delta > 0$), which is obvious since most of the possible transitions are pruned[5]. These degradations increase as K becomes too large, which probably highlights some weaknesses of our current implementation. This can mainly be explained by the fact that the average prior probability of any centroid decreases as K increases. This leads to reduce the number of transitions which are informative enough according to the pruning threshold. Moreover, this phenomenon can become worse by taking into account more unreliable prior probabilities of centroids for high values of K because of the limited size of the training set.

5.2. Decoding of meeting data

Second, preliminary decoding experiments have been carried out on the evaluation set of NIST RT 2007[6] dataset (35K words). We use a two-pass recognition process where word lattices are first generated using "simple" models, leading to N-best lists, with N set to $1,000$. Then, more complex LMs are used to rescore these lists. For the rescoring, we use the RNNLM described in [5] and a 4-gram developed for the AMI system [11]. The RNNLM has been trained on 26.5M words with a $65K$ words vocabulary while the n-gram LM is trained on about a billion words with the same vocabulary. Both models reach the same perplexity on the evaluation set of RT 2007. For the decoding, a WFST is built from the RNNLM based with $K = 512$ and $\Delta = 10^{-7}$ and a bigram LM is derived from the 4-gram LM. The WFST and the bigram LM are about the same size. Perplexities of all LMs on RT 2007 are given in Table 2. Acoustic model is represented by relatively simple HMM/GMM trained using maximum likelihood over PLP features (39 dimensions). The model contains 4.5K tied states with 18 Gaussian mixture components per state. No speaker adaptation is performed in order to keep reasonable run times.

Table 3 reports the word error rates (WER) of the best hypothesis directly after the decoding pass using the bigram LM or the WFST, and after rescoring with the 4-gram LM or with the RNNLM. Additionally, the WERs of the best hypothesis returned without any rescoring, i.e., by using only the WFST and the bigram LM, are given. First, we can notice that WERs are a bit high. This is due to the absence of speaker adaptation. Then, it appears that the WER obtained using the WFST is better than when using the bigram LM since an absolute difference of $0.5\,\%$ is reported, as this was suggested by the perplexities. This is consistent with observed perplexities. Finally, after rescoring, the difference is lesser when using the 4-gram LM and it is even reversed when using the RNNLM. Nonetheless, these results are encouraging since our preliminary implementation of the RNNLM conversion scheme performs already as well as n-gram LMs. We will thus continue experiments.

[5]For instance, for $K = 2$ and $\Delta = 10^{-5}$, $99.946\,\%$ of the transitions are pruned, and, for $K = 1024$ and $\Delta = 10^{-7}$, this number becomes $99.986\,\%$.

[6]http://www.itl.nist.gov/iad/mig/tests/rt/2007/

Table 3: *WERs on the evaluation set of RT 2007 using n-gram LM or RNNLM-derived WFST to generate N-best lists and using n-gram LM or RNNLM to rescore them.*

Rescoring \ Decoding	2-gram LM	WFST derived from RNNLM
No rescoring	47.8 %	47.3 %
4-gram LM	45.2 %	45.0 %
RNNLM	42.9 %	43.2 %

6. Conclusion

In this paper, we have proposed a new strategy to directly exploit probabilities estimated by RNNLMs in the ASR decoder. This strategy consists in converting a RNNLM into a WFST by means of discretization and pruning. We have proposed an original implementation of this generic strategy by using K-means clustering and entropy-based pruning. Achieved results on the Penn Treebank and RT 2007 corpora show that this strategy is promising since the generated WFSTs lead to similar performance to the one of n-gram LMs. Nevertheless, some improvements are still necessary. Especially, a more elaborate pruning criteria could be defined to examine the importance of a transition. However, this task is difficult since estimating the entropy of a RNNLM is complex. Finally, the discretization step could probably also be improved. For instance, it could be interesting to use other possible distances than the default L2 distance to compute the centroids. Measures based on the Kullback-Leibler divergence appear as a natural option towards this objective. Eventually, the employment of hierarchical clustering may also reduce the loss of information due to back-off.

7. Acknowledgements

The authors would like to thank Tomáš Mikolov for providing the RNNLMs used in this paper. This work is funded by the project CTI 12189.2 PFES-ES of the *Comité pour la Technologie et l'Innovation* (Switzerland).

8. References

[1] T. Mikolov, M. Karafiat, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. of Interspeech*, 2010, pp. 1045–1048.

[2] H. Ney and S. Ortmanns, "Dynamic programming search for continuous speech recognition," *IEEE Signal Processing Magazine*, pp. 64–83, 1999.

[3] M. Mohri, F. C. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," *Springer Handbook of Speech Processing*, pp. 559–584, 2008.

[4] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Černocký, "Empirical evaluation and combination of advanced language modeling techniques," in *Proc. of Interspeech*, 2011, pp. 605–608.

[5] S. Kombrink, T. Mikolov, M. Karafit, and L. Burget, "Recurrent neural network based language modeling in meeting recognition," in *Proc. of Interspeech*, 2011, pp. 2877–2880.

[6] A. Deoras, T. M. Mikolov, S. Kombrink, M. Karafiát, and S. Khudanpur, "Variational approximation of long-span language models for LVCSR," in *Proc. of ICASSP*, 2011, pp. 5532–5535.

[7] A. Deoras, "Search and decoding strategies for complex lexical modeling in LVCSR," Ph.D. dissertation, Johns Hopkins University, 2011.

[8] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987.

[9] A. Stolcke, "Entropy-based pruning of backoff language models," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 270–274.

[10] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of English: the Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[11] T. Hain, L. Burget, J. Dines, P. N. Garner, A. E. Hannani, M. Huijbregts, M. Karafit, M. Lincoln, and V. Wan, "The AMIDA 2009 meeting transcription system," in *Proc. of Interspeech*, 2010, pp. 358–361.