Direct Word Graph Rescoring Using A* Search and RNNLM

Shahab Jalalvand , Daniele Falavigna

Human Language Technology unit, Fondazione Bruno Kessler, via Sommarive 18, Trento, Italy
{jalalvand,falavi}@fbk.eu

Abstract

Neural Network Language Models (NNLM) have shown a significant improvement in Automatic Speech Recognition (ASR), however, they are usually too complex to be used directly during trellis search. Although, one could use NNLM for rescoring the N-best list, we propose in this paper a novel method for rescoring directly the hypotheses contained in the word graphs, generated with an ASR decoder. The method, based on the A* search, rescores the partial theories of the stack with a linear combination of the acoustic model score and multiple language models (including NNLM) scores. We compared, on an ASR task consisting of the automatic transcription of weather news, the A* based approach with N-best rescoring and iterative confusion network decoding. Using the proposed method, we measured a relative Word Error Rate (WER) improvement of about 6%, on the given task, with respect to using the baseline system. The latter improvement is comparable with that obtained with N-best list based rescoring method.

Index Terms: neural network language model, word graph, rescoring, confusion network, A* stack search

1. Introduction

The Language Model (LM), usually trained on a large set of text data, allows predicting the a priori probability of a word sequence *W*. In ASR, it is used in combination with the acoustic model (AM) to estimate the maximum a posterior probability of a sequence of acoustic observations.

N-gram based LMs [1] are limited by both the use of discrete probability densities, to compute P[*W*], and the need for implementing back-off procedures to handle n-grams not seen in the training set. In the literature, different smoothing methods have been discussed for discounting and redistributing probabilities [2][3][4], whose performance mostly depend on the size of training data.

More recently, to overcome above shortcomings, continuous space LMs based on: back-propagation neural network [5], neural network exploiting deep learning [6] and recurrent neural network LM (RNNLM) [7], have demonstrated excellent performance to predict the a priori probability of a sequence of words. However, because of the present difficulty to train deep neural network LMs and RNNLMs over large amount of training data they are usually trained on a limited set of text data (e.g. tens of millions of words). This set of data is usually selected to be in a specific application domain, where the ASR system will have to work. In addition, it is difficult to take into account the long-spanning capabilities of both deep neural network LM and RNNLM when the search is carried out over the trellis of acoustic frames. For this reason, a usual approach is that of generating N-best lists by means of a "general-purpose" n-gram LM and rescoring them with the domain-specific RNNLM [8] or deep neural network LM (DNNLM). Nevertheless, N-best list generation is rather complex if N is large and there is no guarantee that the best hypothesis is present in each list. For this reason, some approaches exploiting hill climbing search type over either word lattices or Confusion Networks (CN) has been proposed in [9][10]. Hill climbing search exploits an iterative decoding method for rescoring each hypothesis in a CN or word lattice using the RNNLM probability of a whole sentence hypothesis. However, implementing hill-climbing iterative decoding requires a significant computational effort, especially if the word lattice, or the corresponding CN contains a large number of transitions.

In order to rescore the full search space defined by the word graphs (WGs) and, at the same time, taking advantage from the long-spanning prediction capability of RNNLM, we propose to rescore all of the partial hypotheses contained in the given WG with a linear combination of LMs, including RNNLM. Partial WG hypotheses are generated via A* stack decoding [11][12][13] using as look-ahead function the backward scores of the original WG. We tested the proposed approach on a task consisting of the automatic transcription of weather reports. On this latter we compared performance of A* stack rescoring approach with both N-best list and iterative CN rescoring, both in terms of word error rate (WER) and Oracle Word Error Rate (OWER). The latter is defined as the WER of the best hypothesis contained in the given search space, represented by an N-best list, a word graph or a confusion network.

The rest of this paper is organized as follows. In Section 2 we describe audio and text corpora, used to train the various acoustic models (AMs) and LMs. In the same Section we also give some details on the ASR systems used to generate the search spaces for rescoring. In Section 3 the rescoring approaches used in the experiments, namely: N-best list based method, iterative decoding over CNs and the proposed A* based approach are described in details. Results and experiments are reported in Section 4, and finally, Section 5 concludes the paper.

2. Description of AM/LM training

Within the framework of the European project EU-BRIDGE[1], an ASR evaluation campaign has been organized in order to transcribe weather reports. To do this, a set of training, development and test data have been acquired and delivered to the partners involved in the campaign.

2.1. AM training

Audio training data consists of around 120 hours of weather forecast. A part of these recordings is associated with captions which are not verbatim transcription of the audio itself, but sometimes they are closely related to it.

[1] This work has been partially founded by the European project EU-BRIDGE, under the contract FP7-287658.

A lightly supervised [14] approach was employed in order to train domain-specific triphone Hidden Markov Models (HMMs). The procedure consisted of:

1. Training a set of acoustic models over a large general-domain corpus from EURONEWS channel (about 525 hours).
2. Automatically transcribing the above mentioned 120-hour weather news using the latter acoustic model and an 4-gram LM adapted to the weather-domain text data.
3. Selecting the audio segments where the related automatic transcription and caption matches.
4. Training weather-specific HMMs on the selected speech segments.

The last three steps of the above procedure were repeated two times, resulting in a portion of selected in-domain speech that grew to 57 hours and then to 65 hours. Given the modest improvement in the third iteration, the procedure was not repeated further. In conclusion, the method allowed to automatically select about 55% of the provided training speech, which was considered satisfactory to train domain specific AMs.

Note that in the experiments reported in Section 4, we have compared the performance of the various rescoring approaches using three different sets of HMMs, named: *AM1*, *AM2* and *AM3*, respectively trained on: 1) the general broadcast news corpus (525 hours); 2) on the set of weather audio reports selected after the first iteration of the lightly supervised training procedure (57 hours); and 3) on the portion resulting from the application of the second lightly supervised training iteration (65 hours).

Table 1 gives some statistics on the audio data selected to train the three sets of AMs mentioned above, as well as statistics on development (Dev) and Test sets exploited in the experiments.

Table 1. *The exploited audio and text data*

	Set	≈Hours	≈Running words	Domain
Audio data	Train1 (AM1)	525 hr	3 G	General
	Train2 (AM2)	57 hr	638 K	Weather
	Train3 (AM3)	65 hr	711 K	Weather
	Dev	1.3 hr	12 K	Weather
	Test	1.3 hr	12 K	Weather
	Set	**≈Words**	**≈Vocab size**	**Domain**
Text data	Subtitles	1 G	8 K	Weather
	Auto-sel	100 G	277 K	Related to weather
	Google-news	1.6 T	300 K	General
	Dev	12 K	1 K	Weather

2.2. LM training

As for audio data, a set of weather reports, containing about one millions of words (named Subtitles in Table 1), has been made available. From this latter we trained an in-domain 4-gram LM which was successively used to automatically select documents from a general corpus of news, named "Google-news". This latter is an aggregator of news provided and operated by Google that collects news from many different sources, in different languages. We download daily news from this site, filter-out useless tags and collect texts. Up to now our version of "Google-new" text corpus contains around 1,6 billion words. We ordered documents of “Google-news” according to increasing perplexity (computed with the previously mentioned in-domain LM) and we selected the documents with lowest perplexities to build a corpus of about 100 million words [15], over which a 4-gram back-off LM was trained with the IRSTLM toolkit [16], using the modified Kneser-Ney interpolation method. Then, the latter LM was adapted to the weather domain using the 1MW weather subtitles and the mixture adaptation method as provided by the same IRSTLM toolkit. The latter, mixture adapted LM (hereinafter we will name it as *LMbase*), together with a lexicon, was exploited to build the Finite State Network used during two ASR decoding passes. Then, from the in-domain 1MW corpus, we trained a 4-gram back-off LM, using the Kneser-Ney smoothing method and the SRI toolkit [17]. Hereinafter we will name this LM as *LMin*. Finally, on the same 1MW in-domain corpus we trained an RNNLM with 450 hidden neurons and 1000 classes using the RNNLM toolkit [18]. Table 1 reports also some statistics related to LM training data.

On the weather development set we measured the perplexity values of the various trained LMs, as well as of an additional linearly interpolated LMs: RNNLM+*LMin*. In this case interpolation coefficients were computed by means of the expectation/maximization algorithm with the aim of minimizing the perplexity on the development data. Table 2 gives the obtained perplexity results (more details on the latter evaluation have been published in [19]) and the corresponding Out-Of-Vocabulary (OOV) rates. We point out that PPL value related to *LMbase* in Table 2 has been computed with IRSTLM toolkit [16], while the other PPL values in Table 2 were computed with RNNLM toolkit [18]. Since perplexity is computed differently by the two toolkits, the corresponding values in Table 2 are not directly comparable. Nevertheless, it is worthy to note the PPL improvement gained with the linearly interpolated LM: *RNNLM+LMin*. The latter was used in the rescoring experiments reported below.

Table 2. *Perplexity values (PPL) and Out-Of-Vocabulary (OOV) rate of the different LMs*

LM	PPL	OOV%
LMbase	45.8	0.0
LMin	39.6	0.04
RNNLM	34.7	0.04
RNNLM+LMin	31.6	0.04

2.3. Word graphs generation

The manually detected segments of the development/test set are grouped by a segment clustering method, based on the Bayesian information criterion, then, cluster-wise feature normalization and acoustic model adaptation are applied. The ASR system employs *LMbase*, along with continuous density, state-tied, cross-word, gender-independent triphone HMMs as the acoustic models in a two decoding recognition passes.

Speaker adaptively trained HMMs used in the first decoding pass were trained [20][21] with acoustic

observations obtained through: 1) unsupervised, cluster based normalization of 52 dimensional acoustic feature vectors (to this purpose constrained maximum, likelihood linear regression [22] is used) and 2) Heteroscedastic Linear Discriminant Analysis (HLDA) projection of 52 dimensional normalized feature vectors into 39 dimensional ones. In the second decoding pass, speaker adaptively trained triphone HMMs were trained on normalized, HLDA projected, acoustic features by applying a cluster based, affine transformation estimated w.r.t triphone HMMs, with a single Gaussian density per state, through CMLLR [20][21][22]. In both cases, triphone HMMs were trained through a conventional maximum likelihood procedure. At recognition stage, the output of the first decoding pass is exploited as supervision for CMLLR-based feature normalization and MLLR-based acoustic model adaptation. A frame synchronous Viterbi beam-search is used to find the most likely word sequence.

In the second recognition pass, the decoder generates for each given speech utterance the best word sequence, that was used to evaluate the baseline performance, as well as a word graph. From the latter the list of related N-best hypotheses was computed as well as a confusion network. The graph error rate measured on the utterances of the development set, using the best set of available AMs, resulted to be 4.3%, around 1/3 of the related WER.

3. Rescoring approaches

In this section, starting from simple N-best list rescoring, we go through the state-of-the-art iterative CN decoding approach and we end with our proposed method which is based on A* stack rescoring.

3.1. N-best list rescoring

N-best lists, generated from WGs, provide for each of their ordered hypothesis the related acoustic log-likelihood and baseline LM probability. In this work a new LM score is obtained through linear interpolation of baseline LM probability, RNNLM probability and in-domain LM probability. Then, the hypothesis score, computed summing the acoustic log-likelihood (divided by the LM weight) to the interpolated LM probability, is exploited to reorder each list. In the experiments reported below a value of N=100 was used, while LM interpolation coefficients, as well as the LM weight, were estimated by means of a grid search with the aim of minimizing the WER on the development set.

3.2. Iterative CN decoding

The word graph transitions having a specific amount of time overlap could be merged into a bin. A confusion network [23] is a chain of these bins. All the transitions in a bin are ordered according to their posterior probability. Therefore, the 1-best hypothesis is obtained by concatenation of the first transition in the consequent bins.

Iterative decoding [24] reorders each transition in each bin of a given CN according to a linear combination of scores. For experiments reported below we have used: posterior probability of the transition itself, baseline LM probability of the "locally best" sentence hypothesis (the latter is obtained joining the best left context and the best right context), RNNLM probability of the "locally best" sentence hypothesis and in-domain LM probability of the "locally best" sentence hypothesis. Over a grid of values, we estimated the interpolation weights in order to minimize the WER on the development set.

At each step, just one word is changed and regarding to this change, if the better score is achieved, the change is applied. Therefore, it is guaranteed that after each step a better solution has been found. Like any other hill climbing algorithm, there is the possibility of reaching local minima. To reduce this effect we the simulated annealing algorithm proposed in [9] can be used.

Note that, due to the merging procedure applied when a CN is built, the OWER of a CN decreases in comparison to that related to the corresponding WG or N-best list. At the same time, however, acoustic likelihoods contained in the original WG are missed. This fact prevents direct combination of the LM probability with AM probability for computing hypothesis scores.

3.3. A* stack decoding

A* stack search algorithm [11][12][13] on a WG starts with expanding the first node, pushing the partial paths (sometimes called partial theories) into a stack and sorting them with regard to a total score. The total score is given by the sum of the score of the partial theory and the score furnished by a look-ahead function (see Eq. 1). Here, the look-ahead function of node *i*, *LH(i)* is the total backward score associated to this node (i.e. the logarithm of the sum of the probabilities of all paths starting from *i* reached by the partial path ending in the final state of the WG). Note that the chosen look-ahead function is an admissible heuristic of the future cost of the best path. Hence, for j^{th} partial theory reaching at the node *i* (Th^i_j), the total score S_T is given by:

$$
\begin{aligned}
&S_T(Th^i_j) = S(Th^i_j) + LH(i) \\
&S(Th^i_j) = \frac{AM(Th^i_j)}{\alpha} + \log\left(LM\left(Th^i_j\right)\right) \\
&LM\left(Th^i_j\right) = \beta.LM_{LMbase}\left(Th^i_j\right) + (1-\beta).\left(\gamma.P_{RNNLM}\left(Th^i_j\right) + (1-\gamma).P_{LMin}\left(Th^i_j\right)\right)
\end{aligned}
\tag{1}
$$

where, *AM* is the AM log-likelihood of the theory, P_{LMbase} is the corresponding probability given by *LMbase*, P_{RNNLM} is the probability given by the RNNLM and P_{LMin} is the probability given by *LMin*. The LM weight α, the coefficients β and γ are estimated in order to minimize the WER on the development set. We illustrate the efficiency of the A* stack rescoring approach using a real example given in Figure 1, where a WG is plotted whose reference transcription is "**some sunshine in-between**".

The traditional A* stack search (Figure 1.b), at each step (1, 2, … , 5), popes the top element of the stack, expands the node and pushes the partial theories into the stack and sorts the stack. At any crucial step, like step (3), if the algorithm fails to rank the correct partial theory, because of the bad score given by the baseline LM, there is the hazard of missing the correct theory in the final list. In this example, at step 4, the stack overflows, and therefore, the algorithm must drop the last theory which is actually the correct one. Hereby, we miss the correct solution not only in the 1-best, but also in the final N-best list. To alleviate this issue, we enhance the algorithm in ranking the partial theories by using the scores given by Eq (1).

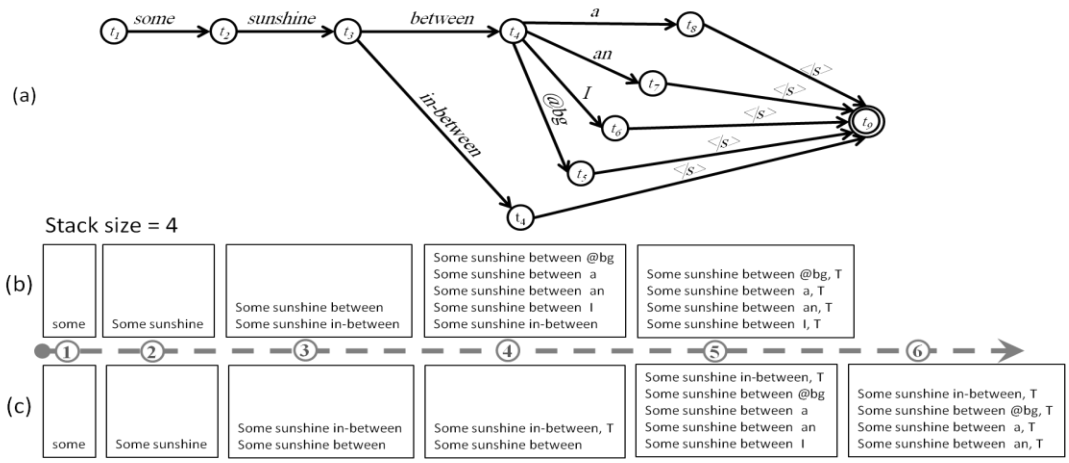

Figure 1. *An example of A* stack search on a word graph; a) a word graph; 2) a normal A* stack search procedure; 3) a rescored A* stack search procedure. (Reference: "**some sunshine in-between**")*

In this way, at the crucial step 3, the algorithm successes in ranking the correct theory and places it on top. At step 4, the top element of the stack is popped, expanded and labeled as a terminator solution. That is, the backward score of this solution is zero, whereby, this theory will have a higher chance to survive in the next steps.

4. Experiments and results

Experiments have been carried out on the given development/test sets using the acoustic models *AM1*, trained on broadcast news domain, *AM2* and *AM3*, trained on weather domain (see Section 2.1). Using each set of acoustic models, along with LM_{base} as the language model, we generate the 1-best, 100-best, WGs and CNs for each utterance in the development/test set, as explained in Sections 2.3 and 3 above. Results obtained with each one of the rescoring approaches described in Section 3 are reported in Table 3. In this Table, the OWER estimated over 100-best "rescored" lists of the development set is also reported. This means that each OWER value reported in Table 3 is computed from a 100-best list generated with the corresponding rescoring method.

Table 3. *%WER and %OWER of baseline systems, using different AMs and rescoring methods.*

AM	Rescoring method	OWER of 100-best Dev	WER on Dev	WER on Test
AM1	Baseline (no rescoring)	-	13.9	12.6
	N-best list rescoring	10.9	13.3 (+4.3)	12.2(+3.1)
	Iterative CN rescoring	10.8(+0.9)	13.7 (+1.4)	12.6(0.0)
	A* stack rescoring	**10.5(+3.6)**	**13.1(+5.7)**	**11.9(+5.5)**
AM2	Baseline (no rescoring)	--	12.2	11.2
	N-best list rescoring	10.1	11.8(+3.2)	10.4(+7.1)
	Iterative CN rescoring	9.9(+1.9)	12.1(+0.8)	11.1(+0.8)
	A* stack rescoring	**9.5(+5.9)**	**11.6(+4.9)**	**10.3(+8.0)**
AM3	Baseline (no rescoring)	--	11.3	10.1
	N-best list rescoring	9	10.6(+6.1)	9.3(+7.9)
	Iterative CN rescoring	9(0.0)	11.1(+1.7)	10.0(+0.9)
	A* stack rescoring	**8.5(+5.5)**	**10.6(+6.1)**	**9.3(+7.9)**

For N-best list rescoring we observe significant improvements with all AMs, and over both development and test set. This is in line with previous works in the literature that used RNNLM for N-best list rescoring.

For iterative decoding with CNs it has to be observed that the CNs include a lot of null bins that dramatically increase the decoding time. Therefore, we decided to remove all bins whose first transition is null (or silence) with a posterior probability higher than 0.99. For using simulated annealing we set the initial temperature as -2000 and the delta as -200.

Looking at Table 3, relative improvements over baseline systems is reached using CN iterative decoding, with all the exploited AMs, although it fails to outperform the other rescoring methods. This is not completely in agreement with previous works in the literature [9][24], however, as mentioned in the introduction, iterative decoding could also be applied to rescore directly the word graphs, e.g., using the methods described in [9][10]. This comparison will be part of future works.

Finally, we observe in Table 3 that the usage of A* based rescoring approach allows achieving slightly better performance, in terms of %WER, than N-best list based approach when the worst acoustic models (*AM1* and *AM2*) are used. Instead, the same %WER (10.6%) of N-best list is obtained when the best acoustic model (*AM3*) is used. Despite this fact, we observe that %OWER improves significantly using A*, i.e. there is still room for decreasing the related %WER, e.g. with RNNLM trained on a bigger set of data.

5. Conclusions

In this paper, we have proposed a new method for rescoring word graphs exploiting RNNLM. We have reported preliminary results showing improvements in performance with respect to the baseline system, N-best list and iterative CN decoding approaches. Much more work still need to be done in order to: 1) show the effectiveness of the proposed method on application domains larger than the one analyzed for this work (transcription of weather reports; 2) estimate the computational requirements of the approach and compare it with other rescoring methods; 3) verify if larger improvements could be achieved with better LMs used for rescoring (e.g. bigger RNNLM, factored LM, exponential LMs, etc).

References

[1] Jelinek, F., "Statistical methods for speech recognition", MIT press, 1997.

[2] Chen, S.F., and Goodman, J., "An empirical study of smoothing techniques for language modeling", Computer Speech and Language, no. 13, pp. 359-394, 1999.

[3] Kneser, R. and Ney, H., "Improved backing-off for n-gram Language Modeling", in proc. of ICASSP, pp. 181-184, 1995.

[4] Gale, W.A., "Good-Turing smoothing without tears", Journal of Quantitative Linguistics, 2vol. 2, pp. 17-237, 1995.

[5] Schwenk, H., "Continuous space language models", Computer Speech and Language, vol. 21 no. 3, pp. 492-518, 2007.

[6] Arisoy, E., Sainath, T. N., Kingsbury, B., and Ramabhadran, B., "Deep neural network language models", Proceedings of the NAACL-HLT, pp. 20-28, 2012.

[7] Mikolov, T., et al. "Recurrent neural network based language model", in proc. Of INTERSPEECH, pp. 1045-1048, 2010..

[8] Sundermeyer, M., Oparin, I., Gauvain, JL., Freiberg, B., Schlüter, R. and Ney, H., "Comparison of feedforward and recurrent neural network language models", in proc. of ICASSP, pp. 8430-8434, 2013.

[9] Deoras, A., Jelinek, F., and Church, K., "Search and decoding strategies for complex lexical modeling in LVCSR", PhD thesis, Johns Hopkins University, 2011.

[10] Rastrow, A., Dreyer, M., Sethy, A. and Khudanpur ,S., "Hill climbing on speech lattices: a new rescoring framework", proc. of ICASSP, pp. 5032-5035, 2011.

[11] Douglas, P.B., "Algorithms for an optimal A* search and linearizing the search in the stack decoder", in proc. of ICASSP, pp. 200-205, 1991.

[12] Douglas, P.B., "An efficient A* stack decoder algorithm for continuous speech recognition with a stochastic language model", proc. of ICASSP, vol. 1, pp. 25-28, 1992.

[13] Knuth, D.E., "The art of computer programming", vol. 2, Addison Wesley, 2011.

[14] Lamel, L., Gauvain, J.L. and Adda, G., "Investigating lightly supervised acoustic model training", in proc. of ICASSP, vol. 1, pp. 477-480, 2001.

[15] Maskey, S., Sethy, A., "Resampling auxiliary data for language model adaptation in machine translation for speech", in proc. of ICASSP, pp. 4817-4820, 2009.

[16] Federico, M., Bertoldi, N., and Cettolo, M., "IRSTLM: an open source toolkit for handling large scale language models", in proc. of INTERSPEECH, pp. 1618-1621, 2008.

[17] Stolcke, A., "SRILM-an extensible language modeling toolkit", in proc. of NTERSPEECH, pp. 901-904, 2002.

[18] Mikolov, T., et al., "Extensions of recurrent neural network language model", in proc. of ICASSP, pp. 5528-5531, 2011.

[19] Jalalvand, S., "Improving language model adaptation using automatic data selection and neural network", in proc. of the Student Research Workshop associated with RANLP, pp. 86-92, Hissar, Bulgaria, 2013.

[20] Stemmer, G., Brugnara, F. and Giuliani, D., "Using simple target models for adaptive training", in proc of ICASSP, vol. 1, pp. 997-1000, 2005.

[21] Giuliani, D., Gerosa, M. and Brugnara, F., "Improved automatic speech recognition through speaker normalization", Computer Speech and Language, vol. 20, pp. 107-123, 2006.

[22] Gales, M.J.F, "Maximum likelihood linear transformations for HMM-based speech recognition", Computer Speech and Language, vol. 12, 75-98, 1998.

[23] Mangu, L., Brill, E. and Stolcke, A., "Finding consensus in speech recognition: word error minimization and other applications of confusion networks", Computer Speech and Language, vol 14, no. 4, pp. 373-400, 2000.

[24] Deoras, A., and Jelinek, F., "Iterative decoding: A novel re-scoring framework for confusion networks." In proc. of IEEE Automatic Speech Recognition and Understanding Workshop, pp. 282-286, 2009.