Parameter Optimization for Iterative Confusion Network Decoding in Weather-Domain Speech Recognition

Shahab Jalalvand, Daniele Falavigna

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

Abstract

In this paper, we apply a set of approaches to, efficiently, rescore the output of the automatic speech recognition over weather-domain data. Since the in-domain data is usually insufficient for training an accurate language model (LM) we utilize an automatic selection method to extract domain-related sentences from a general text resource. Then, an N-gram language model is trained on this set. We exploit this LM, along with a pre-trained acoustic model for recognition of the development and test instances. The recognizer generates a confusion network (CN) for each instance. Afterwards, we make use of the recurrent neural network language model (RNNLM), trained on the in-domain data, in order to iteratively rescore the CNs. Rescoring the CNs, in this way, requires estimating the weights of the RNNLM, N-gramLM and acoustic model scores. Weights optimization is the critical part of this work, whereby, we propose using the minimum error rate training (MERT) algorithm along with a novel N-best list extraction method. The experiments are done over weather forecast domain data that has been provided in the framework of EUBRIDGE project.

Key words: automatic speech recognition, language model, neural network, confusion network, minimum error rate training

1. Introduction

A major problem in domain-specific speech recognition is the lack of sufficient in-domain data for acoustic modeling and language modeling. In the case of language modeling, one could train a n-gram based LM using a huge set of out of domain data and, then, adapt it to the domain using a given set of in-domain data and some adaptation techniques such as the ones described in [1], [2] and [3].

In this paper, we focus on the language modeling part and we introduce efficient approaches for post-processing the output of the automatic speech recognition (ASR) system. The recognizer generates the word graphs for each utterance. Then, we convert them into the Confusion Network (CN) forms. This form yields better oracle word error rate (WER) in comparison to the N-best list and word graphs. Then, we go through an iterative decoding approach for rescoring the confusion networks.

For rescoring the CNs, we adopt an approach similar to the one described by A. Deoras [4], in particular we combine, using iterative decoding, word posterior, RNNLM and NgramLM probabilities. The RNNLM is trained on the small (about 1 million words) set of in-domain data, which consists of captioning of weather forecast news. The reason for using RNNLM is that it has proven to exhibit good performance even if trained on small sizes of training data [5]. In order to estimate the weights to be assigned to RNNLM, NgramLM and posterior probability scores, we utilize Minimum Error Rate Training (MERT) technique [6] along with a novel method for extracting the N-best lists from the CNs.

In Section 2 we will describe the acoustic models and the baseline LM employed in the experiments, as well as the process for generating word graphs and confusion networks. A description of the iterative decoding approach is given in Section 4. In Section 4 we describe the MERT approach developed for learning the weights of the various models used in the rescoring step. Section 5 describes the development/test corpora used and reports the experiments and results. Finally, Section 6 concludes the paper.

1.1. Related Works

The N-gram language model is commonly used in speech recognition systems. Simplicity and low computational complexity are the most important factors of this type of language model which has made it quite popular among the researchers. During the years, different extensions have been made on top of this model to overcome its deficiencies such as data sparseness, generalization and curse of dimensionality. The back-off techniques [7] and the discounting methods [8], [9] are the main extensions over the N-gram LM which are mostly based on making an interpolation between the shorter contexts. However, since in the N-gram LM the words are seen as discrete entities, computing interpolation between their probabilities is, in principle, not possible.

An attempt to change the representation of the words in language modeling was done by Y. Bengio [10], when he introduced the neural network LM. In this model, the words are represented as the binary vectors. Schwenck [11] added a projection layer to the NNLM and named it the continuous space language model. The projection layer converts the binary word vectors into the real number vectors. He also applied this model in a large vocabulary continuous speech recognition system. The probability of the words in these feed forward NNLMs depends on a limited context (the same as the N-gram LMs). T. Mikolov [5] proposed the recurrent neural network LM in which the context is not constrained by a Markov window. The recursive arcs in the hidden layers work as a cache to save the impact of the previous words.

These neural network approaches have shown better performance in terms of Perplexity; however, applying them

directly in the ASR decoder is costly in computation and memory. A common solution is to utilize these models for rescoring the N-best list produced by a traditional ASR decoder which uses a finite state network constructed from a lexicon and an N-gram LM.

However, N-best list rescoring is not the best way to benefit from the high potential of the NNLMs, as the number of the hypotheses limited. For example in our case, the oracle word error rate of the 1000-best list is around 9.9%, while, the word error rate of the 1-best is 10.4%. One could see that there is no big gap in-between. Instead of the N-best list, it is also possible to rescore word graphs or confusion networks. In our case, the oracle word error rate of the word graphs and the confusion networks resulted to be 5.5% and 3.4%, respectively.

2. ASR training and CN generation

For training acoustic models (AMs) we have used audio data provided within the EUBRIDGE consortium containing recordings of weather forecasts. These recordings come with captioning which is not exact transcriptions of the audio so that, in order to train tri-phone Hidden Markov Models (HMMs) a preliminary alignment step is carried out between automatic transcriptions of the training data and the corresponding given captioning. Hence, only the segments of audio recordings that align with the corresponding captioning are retained for HMM training. After this phase about 30 hours of the weather forecasts have been selected for AM training.

For language modeling, we are given a set of weather forecast sentences consisting of about 1 million words. With this latter set of sentences we train an in-domain LM which, in turn, is used for automatically selecting from a large general corpus (see [18]), containing about 1.6 billions of words, the sentences with the lowest perplexity. The automatically selected sentences, formed by about one hundred million words, are used to train a 4-gram, back-off LM which is finally adapted, using the "mix" adaptation method described in [2] to the in-domain data.

From the 4-gram adapted LM, we generate a finite state network (FSN), which also embeds the lexicon, that is used in two ASR decoding passes (the details of the ASR decoder are given in [14]).

Word graphs (WGs) are generated in the second decoding pass. To do this, all of the word hypotheses that survive inside the trellis during the Viterbi beam search are saved in a word lattice containing the following information: initial word state in the trellis, final word state in the trellis, related time instants and word log-likelihood. From this data structure and given the LM used in the recognition steps, WGs are built with separate acoustic likelihood and LM probabilities associated to the word transitions. To increase the recombination of paths inside the trellis and consequently the density of the WGs, the so called word pair approximation [16] is applied. In this way the resulting graph error rate was estimated to be around 33% of the corresponding WER.

Consensus decoding, through confusion network (CN) generation, allows minimizing the word error rate (WER) of sentence hypotheses, instead of maximizing the related posterior probability or, equivalently, minimizing the sentence error rate [15]. A CN is formed by a concatenation of confusion bins, each containing a list of word hypotheses with related posterior probabilities. Basically, a CN is generated from a given WG by: 1) identify CN bins inside the WG corresponding to the non-overlapped time windows, 2) merge all the transitions inside a bin that share the same word (word posterior in a bin is the sum of all the corresponding link posterior in the original WG). In this work, the CNs are produced using the algorithm described in [15] and the software package described in [17].

3. Iterative CN decoding

The method of iterative Confusion Network decoding has already been proposed by A. Deoras [4]. Thus, for further details, we refer the readers to this paper. Here, we briefly describe this method with some variations in our own work.

As mentioned above, a confusion network is a concatenation of bins. The process of iterative decoding, starts from the first bin, re-orders the arcs and shifts to the next one. In each bin, the decoder generates some hypotheses. The number of these hypotheses is equal to the number of the arcs in that bin. Different hypotheses are created by changing a word in the sentence with the words of the bin. Thus, all the hypotheses in each bin differ in just one word. To each hypothesis, the feature functions assign a score. The feature functions, in our case, are RNNLM, N-gramLM, Posterior and Length (the number of the words). The lengths of the hypotheses may differ if there is a null arc in the bin. Then, the scores are interpolated and the resulted score is used to re-order the arcs. After finishing processing a bin, the decoder moves to the next bin and repeats this step. By reaching at the last bin, the score of the best hypothesis (the one which is obtained by concatenating the first arcs) is computed. If this score is better than the one obtained from the previous iteration, the decoder continues this step, otherwise, it stops.

To illustrate the process, we assume a confusion network (*CN*) consisting of four bins (*A, B, C* and *D*):

$$CN : \{A[a_1..a_{n_a}], B[b_1..b_{n_b}], C[c_1..c_{n_c}], D[d_1..d_{n_d}]\}$$

Here, n_a is the number of the arcs in the bin *A*, and so on. Each bin contains a number of arcs and some contents which are assigned to the arcs. These contents are: a word, a posteriori score, an LM score and an acoustic model score. Thus, each arc can be seen as a structure:

$$\begin{cases} a_i.w \rightarrow a\ word \\ a_i.p \rightarrow a\ posteriori\ score \\ a_i.lm \rightarrow a\ language\ model\ score \\ a_i.am \rightarrow an\ acoustic\ model\ score \end{cases}$$

The arcs in each bin are ordered according to their posteriori scores. Hence, the 1-best hypothesis (e^*) in CN is made by concatenating the first arcs:

$$e^* = a_1.w, b_1.w, c_1.w, d_1.w$$

$a_1.w$ is the word assigned to the first arc of the bin *A*. When the decoder starts processing the first bin (A), it will generate n_a different hypotheses:

$$e = \begin{Bmatrix} e_1 = a_1.w, b_1.w, c_1.w, d_1.w \\ e_2 = a_2.w, b_1.w, c_1.w, d_1.w \\ \dots \\ e_{n_a} = a_{n_a}.w, b_1.w, c_1.w, d_1.w \end{Bmatrix}$$

Note that the hypotheses are different in just one word. In order to compare them, we need to compute the new scores.

The RNNLM and NgramLM scores can be computed by applying the LMs on this set of sentences. For the posteriori scores, we can sum up the posteriors of all the words in each sentence or just consider the posteriori of the changing words. Finally, the total score of a sentence is computed by ($i=1..n_a$):

$$\begin{aligned} score(e_i) = \; & \lambda_{rnnlm} \times rnnlm(e_i) + \\ & \lambda_{ngram} \times ngramlm(e_i) + \\ & \lambda_{poster} \times posteriori(e_i) + \\ & \lambda_{length} \times length(e_i) \end{aligned} \quad (1)$$

The length function should be taken into account to avoid being biased towards the short/long sentences. The weights (λ) can be estimated on a development set and by using the optimization techniques.

The critical parts of this method are: selection of the feature functions, and estimation of the weights. In the next section, we describe the MERT algorithm which is a type of machine learning approach for estimating the weights.

4. Minimum Error Rate Training

The MERT algorithm was first introduced by F. Och [6] for using in a statistical machine translation (SMT) task. The algorithm is based on training a parameter model on a set of N-best targets and optimizing the model. The optimized model generates a new set of N-best targets. This set is merged with the one from the previous iteration.

For a reference instane like f_s we aim at finding a candidate in e (that is the corresponding N-best list) which maximizes the total score.

$$\hat{e}(f_s;\lambda_1^M) = \arg\max_{e \in C_s} \left\{ \sum_{m=1}^{M} \lambda_m h_m(e \mid f_s) \right\} \quad (2)$$

In the equation, C_s is the N-best list suggested for f_s. The parameters h_m and λ_m are the function and weight of the m^{th} feature, respectively. In our case, we have four feature functions: RNNLM, N-gramLM, Posterior and length.

The optimized weights for the feature functions can be obtained by solving a minimization problem over the error function $E(r_s,e_s)$.

$$\hat{\lambda}_1^M = \arg\min_{\lambda_1^M} \left\{ \sum_{s=1}^{S} E(r_s, \hat{e}(f_s;\lambda_1^M)) \right\} \quad (3)$$

The value S is equal to the number of the sentences in the development set.

In the extended version of MERT developed by N. Bertoldi et al. [12], the algorithm is run in two loops: the outer loop and the inner loop. Starting from initial weights in the outer loop, the decoder processes the input instances and generates the corresponding N-best list. This list is used to feed the inner loop where the weights are optimized. The inner loop continues optimizing the weights till the time that there is no big change in the weights.

The new weights are again used to run the decoder and generate the new N-best lists. In order to make sure that there is enough diversity among the N-best lists, the new list is combined with the previous one. The outer loop is iterated until the time that no considerable change is observed in WER.

4.1. The M-best Extraction Method

The decoder that is used in our work has been explained in the Section 2.1. The output of this decoder is an N-best list which is extracted from the confusion network. Given a confusion network, one could use a simple A* search algorithm to extract the N-best list from the network. This method that is already embedded in SRI toolkit uses the posterior scores of the arcs in order to output the N-bests. Since, the value of N is limited, the number of the hypotheses will be limited. Therefore, there would be some words in some bins that can never be seen among the hypotheses. It means that, the rescoring process might be again entangled in the lack of hypotheses. This is exactly the problem that is existed with simply rescoring the N-best lists.

In this paper, we propose an efficient method for extracting the candidate list for MERT and we call it "M-best list". In this method, all the possible hypotheses that can be generated in each bin are merged and considered as the N-best list of that step. Therefore, assuming CN as the decoded confusion network, the extracted M-best list includes:

$$e = \begin{Bmatrix} e_1 = a_1.w, b_1.w, c_1.w, d_1.w \\ \dots \\ e_i = a_{n_a}.w, b_1.w, c_1.w, d_1.w \\ e_{i+1} = a_1.w, b_2.w, c_1.w, d_1.w \\ \dots \\ e_{i+1} = a_1.w, b_{n_b}.w, c_1.w, d_1.w \\ \dots \\ e_{i+2} = a_1.w, b_1.w, c_{n_c}.w, d_1.w \\ \dots \\ e_M = a_1.w, b_1.w, c_1.w, d_{n_d}.w \end{Bmatrix}$$

Note that the maximum size of M would be equal to:

$$n_a + (n_b - 1) + (n_c - 1) + (n_d - 1)$$

While, the maximum number of the hypotheses is:

$$n_a \times n_b \times n_c \times n_d$$

The advantages of this method are: 1) the MERT algorithm can see and process all the possible words in its inner loop; 2) there is no boundary for the size of M. According to the size of the confusion network, the number of the sentences could be different, while in the traditional method, this size is limited to N.

The scores of each of these sentences are computed as before. The posterior score of a sentence is also computed by summing up all the posteriors of the words in the sentences.

5. Experiments and Results

In this section, we first describe the details of the corpus that is exploited in this work. Then, we go through the experiments. The reported experiments are arranged as follows:

- Generating the confusion networks on the development and test instances.
- Using Grid search approach for estimating the weights
- Using MERT approach for estimating the weights

We perform these experiments on two sets of confusion networks: one generated using the Bi-gramLM and the other generated using the 4-gramLM.

5.1. The Corpus

The dataset that we have used to analyze and evaluate our approaches is in the domain of weather forecast news,

provided for the EU-BRIDGE project. As mentioned in the Section 2, in this dataset there is an in-domain text set that is around 1 Million words. This data has been used to train the RNNLM and also to select the auxiliary data from the out-of-domain resource. There is also a domain-related text set about 100 MW that has been selected automatically (see Section 2 for the method of selection). The latter set is used to train the Bi-gramLM and 4-gramLM that are used along with the pre-trained acoustic models to generate the ASR output and also the Confusion Networks.

The development and test sets contain 32 and 650 utterances, respectively. The MERT algorithm is run over the development set, in order to estimate and optimize the desired weights for rescoring. Obtaining the optimized weights, the iterative decoding is performed on the test set to rescore the confusion networks.

5.2. Experiments

By using the IRSTLM toolkit [13], we train a Bi-gram and a 4-gram back-off, modified shift beta smoothed language models on the domain-related set (100MW) and we used them in the ASR decoder for generating two different sets of word graphs (one with Bi-gram and one with 4-gram LM). The ASR engine, used for this task is described in [14]. Afterwards, we use the SRI toolkit [17] to convert the word graphs into the confusion networks. At the end, we have two different sets of confusion networks: one created by using the Bi-gramLM and the other by 4-gramLM. The motivation of generating these two sets is to assess the performance of the iterative decoding approach (by the Bi-gram CNs), and improving the results (by the 4-gram CNs).

The confusion networks created in this way contain lots of useless bins with null arcs. This number of useless bins dramatically increases the computational cost. Hence, we filter the confusion networks according to the posterior of the null arcs, i.e. all the bins containing null arcs with higher posterior than 0.99 are eliminated. This filtering decreases the average number of the bins per CN up to 92 percent (without changing the WER).

The resulted CNs yield 16.4% and 10.4% WER on the development set and 20.2% and 14.3% WER on the test set for both Bi-gram and 4-gram CN sets, respectively (see Table 1 and 2).

In order to rescore the confusion networks, we use a RNNLM trained on the in-domain data. The RNNLM is built by the toolkit developed by T. Mikolov, et al [5]. For combining the scores from RNNLM, 4-GramLM, posterior and length, a simple linear interpolation is applied. In order to estimate the weights of these feature functions, we chase two different methods: Grid search and MERT.

For applying the Grid search algorithm, we simply consider an interval from zero to one to assign a weight to each feature function:

$$\begin{cases} \lambda : \{\lambda_{rnnlm}, \lambda_{ngramlm}, \lambda_{posterior}, \lambda_{length} \in [0:0.1:1]\} \\ s.t.\ \lambda_{rnnlm} + \lambda_{ngramlm} + \lambda_{posterior} + \lambda_{length} = 1 \end{cases} \quad (4)$$

By each set of the values, we decode the development confusion networks and the best set is selected to be used on the test set. One could find the results of this method in the second row of the Tables 1 and 2.

Furthermore, we use the MERT algorithm on the development set. In this way, we exploit the proposed method for extracting the M-best lists at the end of each iteration of the decoder. Then, MERT is run to process the M-best list and optimize the weights. On this development set, MERT usually stops at the fourth of fifth iteration. A reason could be the lack of the feature functions. Here, we have just four functions that might not be sufficient. Another reason is the lack of the development data. Nevertheless, in order to validate the weights, suggested by MERT, we ran the algorithm several times on the development set and we selected the best one. The results of this method can be found in the third row of the Tables 1 and 2.

Table 1: The WER results on the confusion networks created by the Bi-gramLM

	Dev	Test
Baseline	16.4	20.2
RNNLM-Grid-ItDec	14.1	18.9
RNNLM-MERT-ItDec	13.5	18.3

Table 2: The WER results on the confusion networks created by the 4-gramLM

	Dev	Test
Baseline	10.4	14.3
RNNLM-Grid-ItDec	10.2	14.3
RNNLM-MERT-ItDec	9.5	14.0

As it can be seen from the tables, the results of the confusion networks created by using the 4-gramLM are apparently better, because the 4-gramLM is more accurate. Note, that the training set and the procedure of training these two LMs are completely the same. Exactly because of the same reason, the improvement in the experiment on the Bi-gramLM is higher. Again, note that the RNNLM used for rescoring both sets of confusion networks is the same. Therefore, one could evaluate the performances of the iterative decoding and the MERT algorithm. Finally, we can see a slight improvement by using the MERT algorithm over the Grid search. It means that the weights suggested by MERT are more efficient than the Grid search. Moreover, the number of iterations taken by MERT is fewer. For example, in MERT, the weights are estimated in 4 or 5 iterations, while for Grid search, we need 66 iterations (according to the intervals considered for the weights in Eq. 4).

There are some deficiencies in the experiments:

- The size of the development set is small and insufficient to have a better weight estimation.
- There are a few feature functions that are not enough for MERT to give a reliable estimation.
- The size of the training set of the RNNLM (1MW) is not comparable with the N-gramLM (100MW).

Considering these deficiencies, we are designing the future experiments, in particular by using more RNNLMs. Due to the complexity of the RNNLM structure, it's not efficient to build it on the big training sets. A wise solution would be to train several RNNLMs on the separated parts of the training set, and then use them as the new feature functions.

6. Conclusion

A set of approaches were introduced and analyzed for improving the process of rescoring the domain-specific ASR output. Instead of the common N-best list rescoring, we used

confusion network rescoring that yields better oracle WER. An iterative decoding approach was used for rescoring the confusion networks and improving the output. Additionally, we applied the MERT algorithm to optimize the weights of the feature functions more efficiently. We also introduced a novel approach for extracting the N-best list from the confusion network that improves the affect of MERT optimization process.

7. Acknowledgement

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

8. References

[1] Federico, M. (1999, September). Efficient language model adaptation through MDI estimation. *In Proceedings of Eurospeech*.

[2] Foster, G., & Kuhn, R. (2007). Mixture-model adaptation for SMT. *In Proceedings of the Second Workshop on Statistical Machine Translation* (pp. 128-135), Prague, Czech Republic

[3] Ruiz, N., Federico, M., & Kessler, F. F. B. (2012). MDI Adaptation for the Lazy: Avoiding Normalization in LM Adaptation for Lecture Translation. *In Proceedings IWSLT 2012*.

[4] Deoras, A., & Jelinek, F. (2009, November). Iterative decoding: A novel rescoring framework for confusion networks. *In proceedings of Automatic Speech Recognition & Understanding (ASRU 2009)* (pp. 282-286).

[5] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. *In proceedings of INTERSPEECH (*pp. 1045-1048).

[6] Och, F. J. (2003, July). Minimum error rate training in statistical machine translation. *In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1* (pp. 160-167).

[7] Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *35*(3), (pp. 400-401).

[8] Witten, I. H., & Bell, T. C. (1991). The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, *37*(4) (pp. 1085-1094).

[9] Chen, S. F., & Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *In Computer Speech & Language*, *13*(4) (pp. 359-393).

[10] Bengio, Y., Schwenk, H., Senécal, J. S., Morin, F., & Gauvain, J. L. (2006). Neural probabilistic language models. *In Innovations in Machine Learning* (pp. 137-186), Springer Berlin Heidelberg.

[11] Schwenk, H. (2007). Continuous space language models. *In Computer Speech & Language*, *21*(3), (pp. 492-518).

[12] Bertoldi, N., Haddow, B., & Fouet, J. B. (2009). Improved minimum error rate training in Moses. *In The Prague Bulletin of Mathematical Linguistics*, *91*(1) (pp. 7-16).

[13] Federico, M., Bertoldi, N., & Cettolo, M. (2008, September). IRSTLM: an open source toolkit for handling large scale language models. *In proceedings of INTERSPEECH* (pp. 1618-1621).

[14] Falavigna, D., Gretter, R., Brugnara, F., & Giuliani, D. (2012, December). FBK@ IWSLT 2012-ASR track. *In Proceedings of the ninth International Workshop on Spoken Language Translation (IWSLT)*.

[15] Mangu, L., Brill, E., & Stolcke, A. (2000). Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *In Computer Speech & Language*, *14*(4) (pp. 373-400).

[16] Ney, H., Ortmanns, S., & Lindam, I. (1997, April). Extensions to the word graph method for large vocabulary continuous speech recognition. *IEEE International Conference on Acoustics, Speech, and Signal, ICASSP-97* (Vol. 3, pp. 1791-1794).

[17] Stolcke, A. (2002, September). SRILM-an extensible language modeling toolkit. *In proceedings of INTERSPEECH*.

[18] Falavigna, D., & Gretter, R. (2012). Focusing Language Models for Automatic Speech Recognition. *In Proceedings of the ninth International Workshop on Spoken Language Translation (IWSLT)*.