Incremental Learning of MAP Context-Dependent Edit Operations for Spoken Phone Number Recognition in an Embedded Platform

Hahn Koo[1] *and Yan Ming Cheng*[2]

Department of Linguistics, University of Illinois at Urbana Champaign, Urbana, IL, U.S.A.[1]
Human Interaction Research, Motorola Labs, 1295 Algonquin Road, Schaumburg, IL 60196, U.S.A.[2]
hahnkoo@uiuc.edu, ycheng@labs.mot.com

10.21437/Interspeech.2006-94

Abstract

Error-corrective post-processing (ECPP) has great potential to reduce speech recognition errors beyond that obtained by speech model improvement. ECPP approaches aim to learn error-corrective rules to directly reduce speech recognition errors. This paper presents our investigation into one such approach, incremental learning of *maximum a posteriori* (MAP) context-dependent edit operations. Limiting our dataset to spoken telephone number recognition output, we have evaluated this approach in an automotive environment using an embedded speech recognizer in a mobile device. We have found that a reduction of approximately 44~49% in speech recognition string errors can be achieved after learning.

Index Terms: error correction, post-processing, speech recognition

1. Introduction

Traditionally, speech recognition research has focused on building better speech models. The most common approaches, derived from HMM theory, simplify speech modeling by making several primitive assumptions (the Markov assumption, the output independence assumption, etc.). Needless to say, tremendous progress has been made to date in spite of these assumptions [1]. However, it is reasonable to speculate that building better models based on such premises places a severe limitation on performance improvement. In order to overcome this limitation, we are shifting our focus away from building better speech models. One direction which bears great potential is error-corrective post-processing (ECPP). Corrective models are derived from errors generated by a state-of-the-art speech recognition system. The resulting model post-processes speech recognition output to further improve speech recognition performance. The fundamental difference between this approach and previous approaches to improve acoustic or language models is that it is not bound by frameworks or assumptions rooted in current models of speech.

In general, ECPP assumes that speech recognition output $\hat{\mathbf{W}} = [\hat{\mathbf{w}}_0 ... \hat{\mathbf{w}}_i ... \hat{\mathbf{w}}_{I-1}]$ is an erroneous version of the truth $\mathbf{W} = [\mathbf{w}_0 ... \mathbf{w}_i ... \mathbf{w}_{J-1}]$, that is, it is received at the other end of an erroneous communication channel. [2] and [3] have attempted to invert the channel similar to the statistical machine translation solution in [4], where $\mathbf{W}$ is considered to be translated from $\hat{\mathbf{W}}$ using a set of machine translation features, such as word translation, word fertility, and word reordering. After training their post-processor iteratively on a speech recognition training set, the authors found that this approach, which predicts a word based on an error-corrected history, leads to a better language model than the approach based on a history that has not undergone error correction. In [5], a $\hat{\mathbf{W}}$-to-$\mathbf{W}$ conversion is realized by applying a series of context-independent word re-ranking transforms at each word position [6]. The transforms are obtained as follows: the word lattice of the speech recognition output is first realigned to form a confusion network; then, word re-ranking transforms are learned at each word position in the confusion network and the position's acoustic and linguistic features by minimizing difference (or error) between the transformed speech recognition output and the truth. In [7], a $\hat{\mathbf{W}}$-to-$\mathbf{W}$ conversion is accomplished by applying a set of context-independent edit operations on $\hat{\mathbf{W}}$; these include insertion, deletion and substitution. The cost of each operation is learned based on the string edit distance [8] in a training set. Then, the edit distance is used as Bayes risk in a minimum Bayes-risk decoding framework [7] to rescore the N-best speech recognition outputs. Significantly, these previous attempts in ECPP rely on context-independent learning in a system based on batch learning during training. In other words, their post-processors are trained on a set of speech recognition errors before deployment.

In contrast, the present strategy assumes that speech recognition errors are better characterized in their contexts. Thus, our post-processor learns to correct errors incrementally using context-dependent information. For some errors, such as those due to imperfect mathematical assumptions and insufficient training data, whether the post-processor is trained before or during deployment matters little. However, because errors may also be due to mismatches between the training and deployment environments and domains, we believe incremental learning is preferable to batch learning for our error-corrective post-processor.

In the next section, we describe our framework of *maximum a posteriori* (MAP) error-edit operations. In section three, we define the incremental learning of MAP edit operations strategy that we employ. The performance of our error-corrective post-processor is evaluated in section 4. As indicated earlier, this evaluation is based on a spoken telephone number task in an automotive environment – which is one of the most significant applications of speech recognizers embedded in mobile devices. Finally, we offer conclusions in section 5.

2. MAP Context-Dependent Edit Operations and Decoder

2.1. MAP context-dependent edit operations

The statistical channel model for ECPP can be generally expressed as $\mathbf{P}(\mathbf{W} \mid \hat{\mathbf{W}})$. A MAP solution of such a system is:

$$\overline{\mathbf{W}} = \arg\max_{\mathbf{W}} \mathbf{P}(\mathbf{W} \mid \hat{\mathbf{W}}) \tag{1}$$

Now, if we assume that $\hat{\mathbf{W}}$ and $\mathbf{W}$ are systematically related, the problem can then be viewed as an exercise in machine translation. In light of this assumption, the above equation can be solved using a greedy search approach [9], where $\hat{\mathbf{W}}$ is an initial guess. This solution searches through a set of very complex modifications on $\hat{\mathbf{W}}$ that include word replacement, fertility, and reordering. This search usually requires many iterations. However, if we assume that $\hat{\mathbf{W}}$ is an erroneous version of $\mathbf{W}$, then a set of simple edit operations (insertion, deletion and substitution) at each word position suffices to make the $\hat{\mathbf{W}}$-to-$\mathbf{W}$ conversion [7][8]. Furthermore, we assume that edit operations are statistically independent of neighboring words in $\mathbf{W}$ and that posterior probabilities are statistically *dependent* in $\hat{\mathbf{W}}$ (context-dependency). Thus, we have:

$$\begin{aligned}
&\max_{\mathbf{W}} \mathbf{P}(\mathbf{W} \mid \hat{\mathbf{W}}) \overset{\text{def}}{=} \prod_{\mathbf{j}} \max_{\mathbf{w}} \mathbf{p}(\mathbf{w}_\mathbf{j} \mid \hat{\mathbf{W}}) \\
&\overset{\text{def}}{=} \prod_{\mathbf{j}} \max_{\mathbf{w_j} \in \mathbf{V}} \mathbf{p}(\mathbf{w}_\mathbf{j} \mid \hat{\mathbf{w}}_\mathbf{i}, \hat{\mathbf{C}}_\mathbf{i}^\mathbf{N}) \\
&\approx \prod_{\mathbf{j}} \sum_{\mathbf{e} \in \mathbf{E}} \max_{\mathbf{w_j} \in \mathbf{V}} \mathbf{p}(\mathbf{w}_\mathbf{j}, \mathbf{e} \mid \hat{\mathbf{w}}_\mathbf{i}, \hat{\mathbf{C}}_\mathbf{i}^\mathbf{N}) \\
&\approx \prod_{\mathbf{j}} \max \left\{ \begin{array}{l} \max\limits_{\mathbf{w_j} \in \mathbf{V}} \mathbf{p}(\mathbf{w}_\mathbf{j}, \mathbf{ins} \mid \hat{\mathbf{w}}_\mathbf{i}, \hat{\mathbf{C}}_\mathbf{i}^\mathbf{N}) \\ \mathbf{p}(_, \mathbf{del} \mid \hat{\mathbf{w}}_\mathbf{i}, \hat{\mathbf{C}}_\mathbf{i}^\mathbf{N}) \\ \max\limits_{\mathbf{w_j} \in \mathbf{V}} \mathbf{p}(\mathbf{w}_\mathbf{j}, \mathbf{sub} \mid \hat{\mathbf{w}}_\mathbf{i}, \hat{\mathbf{C}}_\mathbf{i}^\mathbf{N}) \end{array} \right\}
\end{aligned} \tag{2}$$

where V is vocabulary; _ is a null symbol; **E** is a collection of edit operations: insertion (ins), deletion (del) and substitution (sub); $\mathbf{p}(\mathbf{w}_\mathbf{j}, \mathbf{e} \mid \hat{\mathbf{w}}_\mathbf{i}, \mathbf{C}_\mathbf{i}^\mathbf{N})$ is the joint probability of j^{th} word (or null symbol) in truth and an edit operation, **e**, conditioned on the i^{th} word of the speech recognition output and its context $\mathbf{C}_\mathbf{i}^\mathbf{N}$. The superscript **N** is the parameter of the context window to indicate that the window size is N words to the left and N words to the right at the i^{th} word of the speech recognition output.

2.2. Decoding of MAP context-dependent edit operations

Due to the assumption of statistical independence of edit operations, decoding in (2) can be performed in one pass from left-to-right. However, the drawback of one-pass decoder is that it requires the estimation of a large number of conditional probabilities (at most $(2 \times |\mathbf{V}| + 1) \times |\mathbf{V}|^{2\mathbf{N}+1}$, where $|\mathbf{V}|$ is vocabulary size), which implies a huge storage and search space and more data for robust model estimation. These variables are very sensitive issues for an algorithm which is destined to be implemented in an embedded platform. We therefore improve the decoding process by making two approximations. First, the insertion operation is divided into two operations: (i) inserting a *null* symbol in speech recognizer output $\hat{\mathbf{W}}$, and (ii) substituting the null symbol with a word. Second, the deletion operation is approached by substituting a word with a null symbol in truth $\mathbf{W}$. With these approximations, the set of edit operations is reduced to inserting a null symbol and substituting a word with an expanded vocabulary (original vocabulary plus a null symbol). These modifications allow for an efficient two-pass decoder which we describe below.

In the first pass through the speech recognizer output $\hat{\mathbf{W}}$ from left-to-right, the conditional probability of inserting a null symbol after each word $\mathbf{p}(\_, \mathbf{ins} \mid \hat{\mathbf{w}}_\mathbf{i}, \mathbf{C}_\mathbf{i}^\mathbf{N})$ is evaluated. If the probability is larger than a predefined threshold, a null symbol is inserted after the i^{th} word in $\hat{\mathbf{W}}$. This pass alters the speech recognizer output, denoted $\hat{\hat{\mathbf{W}}}$. In the second pass through the altered speech recognizer output $\hat{\hat{\mathbf{W}}}$, we compute $\overline{\mathbf{w}}_\mathbf{i} = \arg\max_{\mathbf{w_j} \in \mathbf{V}^*} \mathbf{p}(\mathbf{w}_\mathbf{j}, \mathbf{sub} \mid \hat{\hat{\mathbf{w}}}_\mathbf{j}, \mathbf{C}_\mathbf{i}^\mathbf{N})$ at each position. $\mathbf{V}^* = [\mathbf{V}, \_]$ is the expanded vocabulary. Null symbols are then removed to derive the MAP approximation of truth $\overline{\mathbf{W}}$. Consequently, the two-pass decoder has at most $(|\mathbf{V}| + 2) \times (|\mathbf{V}| + 1)^{2\mathbf{N}+1}$ conditional probabilities and a reduced search space. It is worth noting that the possible error(s) of inserting a null symbol introduced in the first pass can be corrected by substituting the null symbol by itself in the second pass. In a practical implementation of the above two-pass decoder, there will obviously be a large number of contexts which never or only rarely appear. To handle both data sparseness and robust estimation, we applied the Laplace estimate of the probabilities. We gave preference to substituting a word with itself, when there is a tie among the probabilities.

3. Incremental Learning of Posterior Probabilities of Edit Operations

Incremental learning of ECPP is an unobtrusive way of constructing a system based on errors observed while the user interacts with the speech recognizer. Initially, the post-processor does not make any change to speech recognizer output. This is because all edit-operation probabilities are the same and preference is given to substituting a word with itself. When a new pair of truth $\mathbf{W}$ and speech recognition output $\hat{\mathbf{W}}$ is given, e.g., collected from the user's manual correction of previous man-machine dialogue results, the pair is aligned word-by-word using dynamic programming. Then, both $\mathbf{W}$ and $\hat{\mathbf{W}}$ are expanded as follows: a null symbol is inserted in $\mathbf{W}$ when deletion is required, while a null symbol is inserted in $\hat{\mathbf{W}}$ when

insertion is required. For example, suppose the speech recognition output is *12234*, while the truth is *12345*. In the speech recognition output, we want one of the *2*s to be deleted and a *5* to be inserted at the end. Therefore, the truth and the speech recognition output are expanded to *12_345* and *12234_*, respectively. Note that after the expansion, both word sequences have the same length and one-to-one alignment. From $\hat{\mathbf{W}}$ and its expansion, we update the probabilities required for insertion in the first pass, either by accumulation or recursion. From the pair of expansions of $\hat{\mathbf{W}}$ and $\mathbf{W}$, we update the probabilities required for substitution in the second pass either by accumulation or recursion. As mentioned above in 2.2, deletion can be approximated by substituting a word in $\hat{\mathbf{W}}$ expansion with a null symbol. Therefore, we do not need to consider deletion probabilities separately in our model.

4. Experiments

The performances of the MAP context-dependent edit operations are evaluated in a spoken telephone number recognition application in automotive environments. The corpus types used in this evaluation are actual US telephone numbers and extensions of variable lengths (4, 7, 10 and 11). These are collected from multiple car types traveling at various driving speeds. SNR varies from 15dB to -5dB due to the adversity of the conditions. The database comprises utterances from 138 users and their corresponding speech recognition output. Each user produced 290 telephone numbers. The speech recognition outputs are generated by a Motorola proprietary embedded speech recognition engine, MLite++. We use a digit loop grammar and speaker-independent acoustic models; the footprint is limited due to the recognizer's embedded implementation. The baseline performance of speech recognition outputs of the database is 80% digit string accuracy (or 96.95% word accuracy). The truth used in the following experiments is database's manual transcriptions.

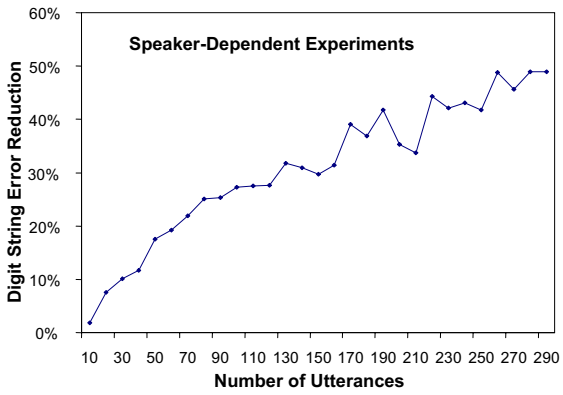

Figure 1 *Average digit string error reduction due to error-corrective post-processing in speaker-dependent experiments with context parameter N=2.*

The first experiment evaluates speaker-dependent performance, i.e., applying our error corrective post-processor per speaker. This speaker-dependent application paradigm is better suited for mobile device applications, since a mobile device is typically used by a minimum number of users, usually one. Figure 1 plots the digit string error reduction percentage averaged over 138 speakers with the contextual parameter *N=2*. As shown in the figure, the error-corrective post-processor is able to correct more speech recognition errors when it learns from more utterances. After 290 utterances, the system reaches 48.9% digit string error reduction.

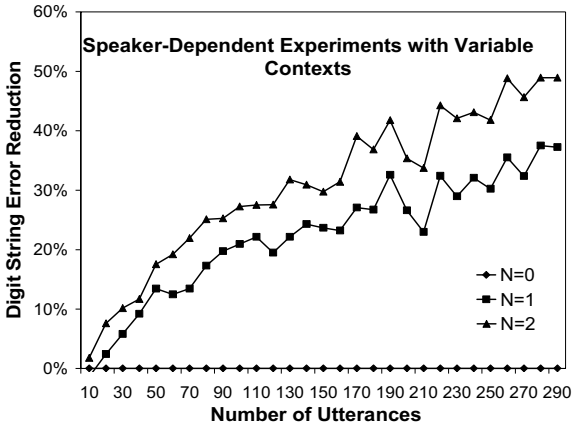

Figure 2 *Average digit string error reduction due to error-corrective post-processing in speaker-dependent experiments while varying the context parameter from N=0 to N=2.*

In the second experiment, we evaluate the effect of contextual information in MAP context-dependent edit operations. Figure 2 plots the average digit string error reductions while varying contextual parameters (*N*) from zero to two. ECPP does not provide any error correction in the context-independent case (*N=0*). However, in both a left and right single word context (*N=1*), it provides a significant jump in terms of string error reduction. Widening the context to two words to the left and two words to the right (*N=2*) improves the performance even more. From these results, we determine that context is an important attribute in such ECPP. There is a trade-off, however: increasing the context leads to a significant increase in memory consumption and requires more data for robust estimation. This trade-off was attested when we further widened the context to *N=3* and *N=4*; average digit string error reduction did not differ significantly from when *N=2*, while memory consumption was significantly increased. We do not elaborate the results of further widening the context to limit our discussion.

To gain further insight, we divided the speech recognition output digit errors into categories according to the edit operations performed, i.e., insertion, deletion and substitution. In Figure 3, we plot the average digit error reduction in each error category. Figure 3 suggests that MAP operations make the most digit error reductions in substitution errors followed by insertion errors. Error reduction in deletion category is virtually zero. This is not very surprising because deletion errors were not very frequent in the database and the two-pass decoder approximated the deletion operation by insertion and substitution.

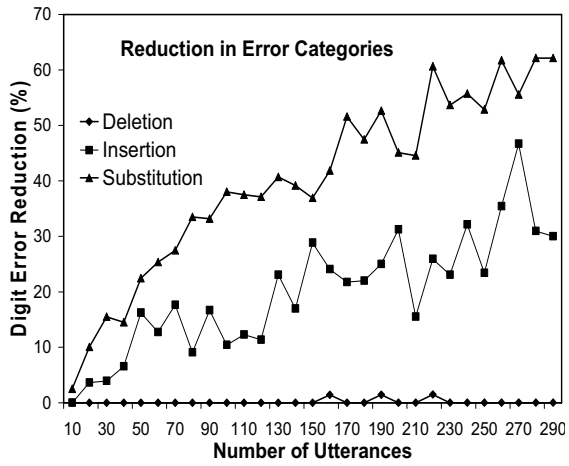

Figure 3 *Average digit error reduction percentage in error categories in speaker-dependent experiments with N=2.*

In our final experiment, we evaluate the performance of the error-corrective post-processor in speaker-independent experiments, i.e., a single error-corrective post-processor is used for all speakers. For this purpose, we randomly ordered samples from all speakers into one giant sequence, and then applied the error-corrective post-processor to the sequence. Figure 4 plots the percentage of digit string error reduction. As in the speaker-dependent experiments, the system is able to incrementally learn MAP edit operations to correct speaker-independent errors. In the end, it can reach a digit string error reduction rate of about 44%. However, it is obvious though that the system learns much more slowly in speaker-independent experiments than in speaker-dependent ones. Moreover, the system occasionally generates errors instead of correcting them in speaker-independent experiments (displayed as negative spikes in Figure 4).

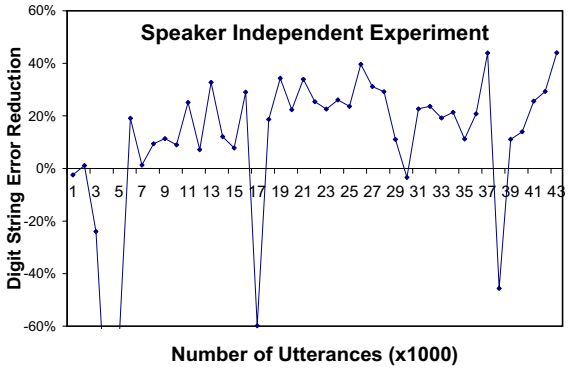

Figure 4 *Digit string error reduction in speaker independent experiment with context parameter N=2.*

5. Conclusion

We have investigated one version of ECPP for speech recognition, specifically, context-dependent error correction incremental model learning. We have introduced a framework of MAP context-dependent edit operation for automatic correction of speech recognition errors and a resource-saving two-pass decoder for embedded implementation. We demonstrated that our error corrective post-processor achieves significant error reduction (44~49%) in spoken telephone number recognition in a very adverse automotive environment, and that the impressive performance gain is chiefly because it relies on context-dependent information. Furthermore, incremental learning allows the post-processor to develop while users are interacting with the embedded speech recognizer. As a consequence, the error corrective post-processor is derived not only from acoustic model errors but also any mismatch between speech recognition training and deployment. Such improvement to speech recognition cannot be currently achieved by simply building a better acoustic model.

6. References

[1] R.K. Moore, "A Comparison of the Data Requirements of Automatic Speech Recognition Systems and Human Listeners," *Proceedings of InterSpeech*, pp. 2582-2584, 2003.

[2] E.K. Ringger and J.F. Allen, "Robust error correction of continuous speech recognition," *Proceedings of the ESCA-NATO Robust Workshop*, 1997.

[3] M. Jeong, J. Eun, S. Jung and G.G. Lee, "An Error-Corrective Language–Model Adaptation for Automatic Speech Recognition", *Proceedings of InterSpeech,* pp. 729-732, 2003.

[4] P.F. Brown, S.A. Della Pietra, V.J. Della Pietra and R.L. Mercer, "The Mathematics of Statistical Machine Translation: Parameter Estimation", *Computational Linguistics*, Vol. 19, No. 2, pp. 263-311, 1993.

[5] L. Mangu and M. Padmanabhan, "Error Corrective Mechanisms for Speech Recognition", *Proceedings of ICASSP,* pp. 29-32, 2001.

[6] E. Brill, "Transformation-Based Error Driving Learning and Natural Language Processing: A case Study in Part of Speech Tagging", *Computational Linguistics*, 21(4): 543-565.

[7] I. Shafran and W. Byrne, "Task-Specific Minimum Bayes-Risk Decoding Using Learned Edit Distance", *Proceedings of InterSpeech,* 2004.

[8] E.S. Ristad and P.N. Yianilos, "Learning String Edit Distance", *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 20(5), pp 522-532, 1998.

[9] U. Germann "Greedy Decoding for Statistical Machine Translation in Almost Linear Time", *Proceedings of HLT-NAACL*, Edmonton, May 2003.