Personalization for BERT-based Discriminative Speech Recognition Rescoring

Jari Kolehmainen1, Yile Gu1, Aditya Gourav1, Prashanth Gurunath Shivakumar1, Ankur Gandhe1, Ariya Rastrow1, and Ivan Bulyko1

1Amazon, US
jkolehm@amazon.com

Abstract

Recognition of personalized content remains a challenge in end-to-end speech recognition. We explore three novel approaches that use personalized content in a neural rescoring step to improve recognition: gazetteers, prompting, and a cross-attention based encoder-decoder model. We use internal de-identified en-US data from interactions with a virtual voice assistant supplemented with personalized named entities to compare these approaches.

On a test set with personalized named entities, we show that each of these approaches improves word error rate by over 10%, against a neural rescoring baseline. We also show that on this test set, natural language prompts can improve word error rate by 7% without any training and with a marginal loss in generalization. Overall, gazetteers were found to perform the best with a 10% improvement in word error rate (WER), while also improving WER on a general test set by 1%.

Index Terms: speech recognition, rescoring, personalization, prompting, gazetteers.

1. Introduction

Production automatic speech recognition (ASR) systems often utilizes two passes, where a first pass generates a list of hypotheses, and a second pass rescores the hypotheses to identify likely transcriptions [1, 2, 3]. For each hypothesis, the second pass models calculate a score (typically, negative log-likelihood) that is combined with the first pass scores to determine the final score [4, 5]. Scores can be computed using a language model [4, 5, 6], a masked language model [7, 8], or other hypothesis-level models [9, 10]. Rescoring performance has been improved by using contextual information in recent literature. This includes audio [5, 11], lattice information from the first pass [6, 12], or personalized context [13, 14].

Common neural rescoring models [15, 5, 7, 9, 10] face challenges with personalized content such as personalized named entities or personal playlists as they cannot adapt to the user’s preferences. In particular, this issue manifests itself for homophone names where neither semantic nor acoustic context can distinguish the correct transcription. Personalized and supplemental n-gram models can address these challenges, but they usually lack generalization and suffer from a specific use case, reducing their effectiveness [14]. Domain classifiers and other auxiliary models can be used to improve generalization properties. This, however, results in a more complex second pass architecture and higher maintenance costs [16]. Including textual context directly in a neural model has been shown effective in many tasks such as document classification [17], language modeling [18], and acoustic modeling [19]. In this study, we demonstrate how personalized named entities can be incorporated into a neural rescoring pass.

The literature provides a wide variety of neural modeling approaches that utilize both unstructured and structured contextual information: techniques that use encoded context via a cross-attention framework [17, 20, 19]; retrieval-based approaches [18, 21]; or approaches that modify the model input via embedding augmentation [22, 23, 24]. For the last category, it can either be achieved by augmenting token-level input embeddings [22, 24] with gazetteers. Alternatively, additional context tokens can be processed as part of the input sequence following certain logic [23, 25, 26, 27]. The latter approach is referred to as prompting. Prompt embeddings can be trainable [23, 25] or discrete optimization can be performed on the existing embeddings to find optimal prompts, without the need to apply conventional model training [26].

There are auto-regressive rescoring models [4, 5, 6] that can be used for rescoring, but bidirectional discriminative models have recently shown superior performance [9, 10]. In this paper, we use a bidirectional transformer model, called RescoreBERT, as the baseline [10]. RescoreBERT is a rescoring model that predicts a single score for each hypothesis and is trained with a discriminative ASR loss, which directly minimizes the word error rate. The model architecture is summarized in Section 2.1 and illustrated in Fig. 1a.

To investigate personalization in neural rescoring, we use internal en-US data supplemented with each user’s personalized named entities. We focus on three novel rescoring approaches: 1) gazetteers where we add trainable slot embeddings to the vanilla RescoreBERT model; 2) natural language prompting where we include prompts to contextualize the RescoreBERT model; and 3) an encoder-decoder architecture with cross-attention to personalized named entities. We show results for these models in different data scenarios, model sizes, and test sets. This study intends to demonstrate the level of improvement each approach can yield in recognizing personalized content and entities without compromising general recognition performance.

2. Proposed Approaches

2.1. Rescoring Model

We use the recently proposed RescoreBERT as the baseline rescoring model [10]. RescoreBERT model uses a pre-trained BERT model with a feed-forward network attached to the classification (CLS) token embedding to predict a single score for each hypothesis (see Fig. 1a). Unlike auto-regressive language models, RescoreBERT scores do not follow a proper probability distribution. In more formal terms, the final score is calculated...
The rescoring model is trained using a standard minimum word error rate (MWER) loss calculated on n-best hypotheses from the first-pass: loss \( L \) [28, 29, 30]:

\[
L = \sum_{i} (\epsilon_{i} - \tau)p_{i},
\]

where \( \epsilon_{i} \) is the edit distance of \( i^{th} \) hypothesis; \( \tau \) is the mean edit distance of all hypotheses originating from the same utterance; and \( p_{i} \) is the normalized posterior distribution of the \( i^{th} \) hypothesis in the n-best, given by the following softmax:

\[
p_{i} = \frac{\exp(-v_{i})}{\sum_{k} \exp(-v_{k})}.
\]  

2.2. Prompts

We consider a setting where each utterance has a set of tokenized strings \( D \) representing personalized named entities. In the prompt approach, RescoreBERT input is concatenated with a natural language prompt if any sub-string (continuous subsequence of tokens) in the hypotheses matches an entity in the set \( D \). We call this a match condition. For example, if \( ['entity'] \) is among the personalized contact names for a particular hypothesis “call ['entity']”, then the match condition is met. In this study, we did not perform any prompt tuning or prompt engineering and instead used a simple phrase “as i need to contact ['entity']” as the augmented prompt. When there are multiple matched entities within a hypothesis, “and” is used in the prompt to separate them.

2.3. Gazetteers

For gazetteers, we process the input tokens \( x_i \) by tagging each token that matches an element in \( D \). Only full matches are tagged and partial matches are ignored. The result is a binary sequence \( y_i \), in which zeros indicate a non-match and ones indicate a match. Note that \( y_i \) has the same length as the input tokens \( x_i \). An embedding layer can convert the slot tags \( y_i \) into dense representations, with tagged tokens (i.e. \( y_i = 1 \)) generating a learnable embedding and non-tagged tokens (i.e. \( y_i = 0 \)) generating a fixed non-learnable zero embedding. We consider two gazetteer variants: early fusion and late fusion, which differ in how the slot embeddings are added to the model.

In early fusion, the slot embeddings are added before the first transformer layer into the token and positional embeddings (see Fig. 1b). The resulting score is computed by

\[
s_{i} = f \circ g(E_{i}(x_{i}) + E_{p}),
\]

where \( E_p \) are the slot embeddings. Note that the slot embeddings are added to the existing embeddings instead of replacing them. This will allow model to learn to distinguish salient words (i.e. “will”) from actual names.

In late fusion, slot embeddings are added to the last transformer layer (see Fig. 1c), the score is computed using:

\[
s_{i} = f \circ (E_{s} + g_{n-1} \circ \cdots \circ g_{1}(E_{1} + E_{p})).
\]

The slot embeddings \( E_s \) are zero when none of the tokens are tagged, and the score is the same as the baseline RescoreBERT in Eq. (1), for both fusion variants.

2.4. Cross-Attention Based Encoder-Decoder

The last personalization approach we considered combines prompting and gazetteers ideas using a cross-attention architecture, as illustrated in Fig. 1d. We use string-matching as
before to find all the matching entities from hypotheses that are in $D$. The matched tokenized personalized entities are concatenated with a classification token (CLS) and separated by separation token (SEP) to form slot token sequence $z_i$. Hence, $z_i$ will appear as [CLS] [entity #1] [SEP] [entity #2] [SEP] · · · [SEP]. The slot tokens $z_i$ are then encoded using the same encoder as the hypothesis input tokens $x_i$. Both the hypothesis and the encoded slot embeddings are used as inputs to a non-causal transformer decoder layer to extract a classification token for the scoring feed-forward network. This is more formally expressed by

\[
\begin{align*}
X_i &= g(E_i(x_i) + E_p) \\
Z_i &= g(E_i(z_i) + E_p) \\
s_i &= f \circ m(Z_i, X_i),
\end{align*}
\]

where $m(\cdot, \cdot)$ is a decoder layer without a causal mask where the first entry represents the keys and values of the multi-head cross-attention and the second entry is for the query (see Fig. 1d). If there are no matching name, the $z_i$ is a single classification token. In this study, the decoder layer used the same hyper-parameters as the encoder (e.g. number of attention heads, hidden size, etc.).

### 3. Experiments

#### 3.1. Data

We use de-identified far-field en-US datasets gathered from interactions with a virtual voice assistant in this study. Training data is split into two parts: The first part consists of personalized utterances where the ground truth transcription contains a reference to a personalized named entity; a second part consists of randomly selected utterances from usage traffic that may not contain references to the personalized entities of users. For example, the personalized utterances could look like “alexa make a phone call to john doe”. These two data sources are combined to create the final training data set. Samples from the personalized data source are added until a predetermined "personalized" fraction is reached. Users typically have a few thousand records of personalized "contact name" entities, each with a first and last name.

#### 3.2. Models

##### 3.2.1. First Pass Model

We use a recurrent neural network transducer (RNN-T) based first pass model that also employs neural personalization [19] and is supplemented with a shallow fusion n-gram model. The first pass score consists of two parts:

\[
u_i = \gamma \log P_{CA-RNN-T} + \epsilon \log P_{SF},
\]

where $P_{CA-RNN-T}$ is the contextual-aware RNN-T model [19] probability; $P_{SF}$ is the shallow fusion n-gram probability; $\gamma$ and $\epsilon$ are interpolation weights. The context-aware RNN-T can improve the word error rate up to $\sim 35\%$ and the shallow-fusion by additional $\sim 5\%$ compared with a standard RNN-T model [19].

##### 3.2.2. Rescoring Models

We use two different rescoring BERT models in this study: (i) tiny BERT; and (ii) big BERT. Both BERT models use a sentence-piece tokenizer with a multilingual vocabulary of $\sim 153k$ tokens (trained on the MC4 dataset [31]). Roughly $\sim 30k$ of these tokens are from English language. The hyper-parameters for the model architecture are outlined in Table 2.

### 3.3. Training and Evaluation Protocol

All the models used the public hugging-face implementation of BERT [32] (model parameters and tokenzier are different). DeepSpeed [33] was used for training and evaluating all models. Batch size was 256 and 128 for tiny and large BERT models, respectively. All experiments used a learning-rate decay policy and an Adam optimizer with standard parameters. The initial learning rate was $10^{-5}$ for the tiny BERT experiments and $10^{-6}$ for the big BERT model experiments. An early stopping strategy, based on the personalized validation set, was applied (see Table 1).

### 4. Results and Discussion

Relative word error rate reduction (WER) numbers using vanilla RescoreBERT models, compared to a baseline without a rescoring pass, are shown in Table 3. Using Oracle rescoring (e.g. picking the hypothesis with the smallest edit distance from truth) improves the first pass significantly, demonstrating the bound on WER improvements on both test sets.

RescoreBERT (see Fig. 1a) improves on the general test set, but degrades the personalized test set. This is likely due to the fact that the first pass is already context aware, while the RescoreBERT models are not. The larger RescoreBERT amplifies this characteristic by causing a more pronounced degradation on the personalized test set. A fine-tuned RescoreBERT model reduces degradation but fails to match the first-pass WER on the personalized test set.

By applying natural language prompts without any training to RescoreBERT models (see Table 4), WER improves significantly over the first pass with some degradation on the general test set. We hypothesize that this is particularly effective for the
Table 3: Baseline WERR results of rescoring models in the personalized and general test sets relative to the first pass (no rescoring).

<table>
<thead>
<tr>
<th>Model</th>
<th>Personalized</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>-57%</td>
<td>-58%</td>
</tr>
<tr>
<td>Tiny RescoreBERT</td>
<td>+3.9%</td>
<td>-5.3%</td>
</tr>
<tr>
<td>Big RescoreBERT</td>
<td>+4.8%</td>
<td>-7.1%</td>
</tr>
<tr>
<td>Tiny RescoreBERT (fine-tuned)</td>
<td>+2.5%</td>
<td>-5.3%</td>
</tr>
<tr>
<td>Big RescoreBERT (fine-tuned)</td>
<td>+1.7%</td>
<td>-6.8%</td>
</tr>
</tbody>
</table>

Table 4: Rescoring model WERR results in personalized and general test sets. Upper section: WERR relative to tiny RescoreBERT. Lower section: WERR relative to the big RescoreBERT. Negative sign corresponds to WER reduction and positive sign WER increase. Percentage sign in brackets in the model name denotes personalized training data fraction.

<table>
<thead>
<tr>
<th>Model</th>
<th>Personalized</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untrained Prompt</td>
<td>-7.1%</td>
<td>0%</td>
</tr>
<tr>
<td>Frozen Late</td>
<td>-4.5%</td>
<td>0%</td>
</tr>
<tr>
<td>Frozen Early</td>
<td>-4.9%</td>
<td>0%</td>
</tr>
<tr>
<td>Trained Late</td>
<td>-13%</td>
<td>+3.9%</td>
</tr>
<tr>
<td>Trained Early</td>
<td>-13%</td>
<td>+3.8%</td>
</tr>
<tr>
<td>Trained C-A</td>
<td>-14%</td>
<td>+5.1%</td>
</tr>
<tr>
<td>Train Early (1%)</td>
<td>-12%</td>
<td>+3.0%</td>
</tr>
<tr>
<td>Fine-tuned Prompt (1%)</td>
<td>-13%</td>
<td>+0.4%</td>
</tr>
</tbody>
</table>

RescoreBERT model. The first-pass model penalizes longer utterances by assigning higher scores. Due to MWER training, the rescoring model learns to counter this by predicting a high-ranking score for longer utterances. By increasing the length of the hypothesis with adding semantically correct prompts, we believe that the hypothesis will receive better scores, thus getting ranked better than other hypotheses. We tested this anecdotally by trying out a variety of semantically correct and incorrect prompts. When trained with personalized data, all the approaches showed the “frozen model” gazetteers perform worse than prompting, which does not require training.

Fig. 2 shows training results for the models when general data is mixed into the training data. Using data mixing effectively mitigates the general test set degradation without significant loss of accuracy on the personalized test set with the exception of the cross-attention approach. The cross-attention model WER on the personalized test set became significantly worse when no general test set degradation was allowed. In the personalized test set, early fusion gazetteer performed slightly worse than cross-attention, but was less sensitive to data fraction. The early fusion model was also the only one that improved WER on the general test set over the baseline RescoreBERT, while also achieving over 10% WERR on the personalized test set.

The bottom half of Table 4 presents the results of repeated experiments using the big BERT model. Prompting with the big RescoreBERT model proved very effective for personalization and provided a similar level of improvement as the trained models. However, it caused a larger degradation on the general test set than prompting with the tiny RescoreBERT. The trained early fusion gazetteer results improved in every aspect using a larger transformer.

5. Conclusions

We compared three different approaches to include personalized named entities in rescoring: (i) early and late fusion gazetteers; (ii) prompts; and (iii) cross-attention based encoder-decoder model. The effectiveness of these approaches was explored using internal data with personalized named entities. When trained with personalized data, all the approaches could deliver over 10% WER improvement over the baseline RescoreBERT model for the personalized test set. However, this leads to degradation in the general test set. Mixing some fraction of general data did address this issue for prompting and gazetteers with small loss of performance in the personalization test set. Cross-attention was slightly more sensitive to the data fraction and we did not find a data mixing ratio that would yield good performance in both test sets. We also found that prompting gave 7% improvement over the baseline WER without any training. This could be particularly useful for cases where training data is limited or not available. Gazetteers proved to be the best in general approach as it was the only model that could give over 10% improvement in the personalization while improving the general data test set at the same time. Both the gazetteers and prompting approaches performance become better with larger model size suggesting that they are scalable.
6. References


