REC-ACE: Robust Error Correction for ASR using Confidence Embedding

Dan Botchan (Dan.Botchan@post.runi.ac.il), Sharon Koubi (Sharon.Koubi@post.runi.ac.il)

Submitted as final project report for the NLP course, Reichman University, 2023

Abstract

Automatic Speech Recognition (ASR) technology has revolutionized various applications by enabling the conversion of spoken language into text. However, ASR outputs often contain errors due to challenges posed by accents, environment ambient noise, and the inherent complexity of spoken language. These errors can affect the usability of the transcribed text in downstream tasks, prompting the need for effective ASR Error Correction and Detection techniques [1].

In this paper, we build upon the impressive results of RED-ACE [2] in the domain of ASR Error Detection (AED). Specifically, we explore the integration of auxiliary Confidence Scores alongside the transcribed text to enhance the performance of the ASR Error Correction (AEC) model. We show that by leveraging these supplementary signals, we bolster the accuracy of AEC in rectifying errors in transcription. Through this research, we aspire to contribute to the advancement of ASR technology, addressing its limitations and paving the way for more resilient and accurate systems capable of excelling in real-world scenarios.

Project Code: https://github.com/SharonKIDC/REC-ACE

1 Introduction

In today's era, the use of Automatic Speech Recognition services has increased significantly [1, 2]. The need to rely on these services is a result of the complex challenges associated with training such models on your own, a process that requires extensive amounts of data and significant computational capabilities. However, despite the potential of these convenient services to overcome the aforementioned challenges, they are typically employed as black-box solutions. Consequently, fine-tuning them for specific datasets is usually not possible. This may lead to inaccuracies when applying them to out-of-distribution data, such as different accents and low-quality audio [1, 2]. Therefore, their direct utilization in downstream tasks that demand high accuracy could be problematic. As a response to this challenge, a novel category of models has emerged, referred to as ASR Error Detection (AED) [1]. These models engage in post-processing of the ASR's output, aiming to identify transcription errors and preemptively notify users prior to the propagation of these errors. The ASR services themselves, also make endeavors to spot errors in the output, by introducing an extra output referred to as Confidence Scores (CS), wherein the ASR model, or an additional model, assigns a score to each transcribed word. This score reflects the level of confidence the ASR model has in accurately transcribing the

spoken word [3].

Drawing from prior research demonstrating enhancements in error detection by incorporating CS as an extra signal, RED-ACE presents a straightforward technique for integrating CS within an adapted BERT model. This approach effectively combines the advantages of both the Transformer architecture and the supplementary signal, yielding improved outcomes in AED. In our project, our aim is to assess the results obtained by RED-ACE in AED, within the domain of ASR Error Correction. AEC focuses on enhancing the accuracy of transcribed text by rectifying these inaccuracies. Employing a combination of methods that scrutinize the ASR output, identify misinterpretations, and propose corrections that align with the context and language rules. Nonetheless, the issue of error correction remains an ill-posed challenge, as the greater the inaccuracy within a transcription, the more diverse the potential corrections become. Utilizing the RED-ACE approach, we intend to incorporate CS as supplementary input into a sequence-to-sequence oriented Transformer model, aiming to enhance output sentence refinement by relying on the prior semantic and contextual understanding of a T5 model [4], and the auxiliary information given the confidence scores, as for where are the potential errors that require correction. We denote this model as RE**C**-ACE: "Robust Error Correction for ASR using Confidence Embedding" and show results in comparison with a baseline T5 Model, that was trained solely on the transcription hypothesis, and another comparison with a T5 Model that was trained to embed the CS signal through its prompts.

2 Background

Despite significant advancements in ASR technology, high error rates in out-of-distribution domains, continue to hinder the adoption of this technology. The three common types of errors in ASR are: 1. Substitution - a word might be changed to a different word when it's transcribed. 2. Deletion - a word from the reference might be left out in the transcription hypothesis. 3. Insertion- sometimes a new word appears in the transcription hypothesis that wasn't in the reference [1].
Error correction involves taking the initial transcription hypothesis and modifying it to be similar to the original spoken sentence. Certain approaches, break the task into two parts: error detection and proposing corrections based on the identified errors [5]. Contemporary approaches, often train sequence-to-sequence models. Frequently involves adaptations of the Transformer encoder-decoder architecture [6, 7]. More recently, Large Language Models (LLMs) have also been employed for this task. They aim to use their existing knowledge of the world, to fix the ASR hypothesis through prompts within the LLM [8].
Another distinction between these methods lies in the input signals they utilize. While they all make use of the transcription hypothesis, some incorporate additional signals like multiple transcription hypotheses and ASR lattices [9] or phonetic information of an entity [10]. Our project builds upon the foundation laid by RED-ACE [2], which introduced the incorporation of Confidence Scores into the BERT model through an encoder modification involving an additional embedding layer. Although their work primarily focuses on ASR Error Detection, we aim to demonstrate, as they proposed, that this approach is equally advantageous for the task of Error Correction.

3 Dataset

For the sake of training and assessing their AED models, RED-ACE [2] curated a dataset containing annotated transcription errors. They leveraged the LibriSpeech corpus [11], which encompasses 1000 hours of English speech transcriptions. By employing a Google Cloud Speech-to-Text API as their ASR model, they derived hypothesis transcriptions and Confidence Scores from the corpus. Within their dataset, there exist two distinct categories of subdatasets, labeled as "clean" and "other". These labels correspond to the audio quality of each subdataset. Notably, the "clean" category exhibits superior recording quality in contrast to the "other" category. In our study, we will utilize a set of triplets com-

prising Hypothesis Transcription, Confidence Scores, and Ground Truth transcription, from the RED-ACE dataset. The former two will serve as input, while the latter will guide the training process as the supervision signal.

4 Method - REC-ACE

Building upon the research conducted by RED-ACE [2] which modified BERT, we employed a pretrained T5 model harnessing its prior semantic and contextual knowledge from training on multiple tasks [4, 9]. We reconfigure the T5 encoder to incorporate confidence scores, refining it through fine-tuning to execute corrections on transcribed sentences from the ASR model. Our new error correction model, which we named REC-ACE, uses the ASR's hypothesis transcriptions and their associated confidence scores to predict sentences most probable to have been spoken during the speech-to-text conversion. In the following subsections, we will explain how it works.

4.1 Confidence Scores Binning

The integration of CS is executed following the approach suggested by RED-ACE [2]. We start by converting the CS vectors from floating-point to integers through the application of a binning algorithm. RED-ACE has shown that the binning algorithm effect on results is negligible.
Therefore, we simply divide the range of the scores $[0, 1]$ into B bins. Such that $D_i = \{c_j \mid b_{i-1} \leq c_j \leq b_i\}$. Where $D_i, \forall i \in \{1, ..., B\}$ is the integer we assign to the score c_j s.t $j \in \{1, ..., N\}$ is the j-th score in N words sentence. Following binning, the new vectors represented as integers can be interpreted as input tokens to the new embedding layer.

4.2 Confidence Scores Embedding

In its fundamental form, the T5 encoder receives word tokens as inputs. These word tokens undergo conversion into embedding through an embedding layer, which essentially functions as a lookup matrix denoted as $M_{words} \in \mathbb{R}^{V \times H}$. Where H is a hyperparameter, represents the hidden vector's size, and V corresponds to the tokenizer's vocabulary size. In accordance with the REC-ACE study, we adapt the encoder by incorporating an additional embedding layer for the embedding of the confidence scores. The updated embedding layer can be represented as lookup matrix $M_{CS} \in \mathbb{R}^{B \times H}$, where B denotes the number of bins in the binning algorithm, and H maintains the same dimensions as the hidden vector size for words. The new encoder initially maps the word tokens and the CS tokens into a common $\mathbb{R}^H$ space, subsequently combining the two embedding vectors through addition, and then routing the amalgamation to the remaining blocks of the T5 model, as can be seen in Figure 1.

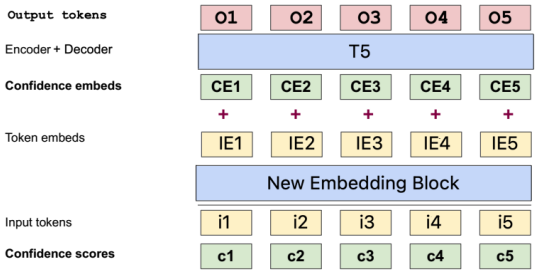

Figure 1: A confidence-aware ACE model. A T5 for conditional generation was used with the modification of a new embedding block, which contains an additional embedding layer for the quantized confidence scores

The decoder component of T5 remains unchanged and is responsible for generating coherent and contextually relevant text.

5 Experiments

Our experiment seeks to assess the potential effectiveness of CS scores in the error correction domain, and whether we can mirror their impact in the error detection domain. To assess this, we will evaluate a T5 model fine-tuned both with and without the supplementary Confidence Scores signal to see whether it has an impact on the metrics evaluated. Addition-

ally, we will explore the feasibility of integrating CS as supplementary cues through prompt engineering. This involves amalgamating the two signals into a single prompt and subsequently training another T5 model based on the suggested prompt.

5.1 Experiments Settings

First, our intention is to employ a pretrained T5 model, capitalizing on the accumulated semantic understanding garnered during its training across diverse NLP tasks [4]. In this manner, the model might be able to draw upon its acquired knowledge of the world to construct a coherent and meaningful sentence correcting the ASR transcription. This evaluation will be conducted through the process of finetuning a preexisting T5 model on our dataset, followed by a comparison with the outcomes derived from the original ASR output. Then, the additional T5 model undergoes modification to introduce a fresh supplementary embedding layer to handle the confidence scores vector input. The adapted T5 model, REC-ACE, can undergo evaluation in comparison to the aforementioned fine-tuned model. Finally, we will integrate the transcription hypothesis and the CS signal into the T5 Model through "prompt engineering" to see if the new embedding layer is necessary. We will merge both signals into a unified prompt and subsequently finetune the model through instructional guidance to make corrections based on this new prompt.

The Prompt:
"Correct the sentence based on confidence scores: sentence: {words_vec}, scores: {scores_vec}"

Each of the models will undergo assessment using the two distinct subsets: "clean" and "other," analogous to the approach taken by RED-ACE as outlined in their paper. This evaluation aims to ascertain the models' resilience when confronted with varying audio quality conditions. Specifically, every model will undergo training twice—once on the "clean" subset and once on the "other" subset. Subsequently, each configuration will be subjected to testing across both subsets to comprehensively evaluate their performance.

At this phase, an interesting observation arises: training on the "other" dataset appears to be more stable compared to the "clean" dataset (can be seen in *train_pipeline* notebook). This trend could potentially signify that the "other" dataset contains a more diverse range of error examples. Therefore, the model demonstrates improved generalization capabilities on both "other" and "clean" dev sets. Subsequently, It will be interesting to see if this results will repeat on the test set.

5.2 Baselines

Up to this juncture, we haven't identified any other papers that utilize the dataset published by RED-ACE [2]. Thus, we first evaluate the metrices between the ASR transcription hypothesis and the reference text to get baseline results, then we compare our REC-ACE model against a fine-tuned T5 model as mentioned in Sec 5.1, to test whether the additional signal improves results. An additional assessment will determine whether the additional embedding layer is necessary or if we can integrate the CS signals by introducing instructions to the T5 model. This approach capitalizes on the fact that T5 has been trained with instructions.

5.3 Tokenizer

We were faced with the decision of choosing between employing a predefined word tokenizer or constructing a new one based on our dataset corpus. The first option could result in many words being OOV, but the predefined tokenizer is based on SentencePiece [12], which operates as a sub-word tokenizer. This capability allows it to break down OOV words into smaller fragments present in its vocabulary. Thus, we can use the pretrained words embedding layer, of the T5 model, to utilize the available computational resources effectively.

5.4 Training

During the training phase, we finetune all model parameters on REC-ACE dataset based on their splitting into train, dev & test sets. We chose to finetune all the parameters, operating under the assumption that the additional information from the CS embedding will harmonize within the model's di-

verse layers. This finetuning occurs in a supervised, end-to-end fashion, wherein the model's predicted tokens are compared with the label tokens.

To initiate training, we chose a pretrained "t5-small" model with 60.5M parameters obtained from HuggingFace. The finetuning process occurs on an Nvidia GeForce RTX3090 GPU, utilizing a batch size of 64. We used ADAM optimizer with a learning rate of $1e^{-4}$ and applied CrossEntropy as our loss function for 50 epochs. We used the WER score from each epoch, to determine which is the best checkpoint, for further evaluation. The training process for each scheme required approximately 12 hours.

The training pipeline is outlined in the *train_pipeline* notebook.

5.5 Evaluation Metrics

In contrast to ASR Error Detection, for which each word is labeled as Error or Not Error, and one can use common classification matrices such as precision, Recall, and F1 [2], ASR Error Correction models are a text-to-text type of models. Therefore, their evaluation often encounters the typical obstacle of being an ill-posed problem. The input sentences for the models are corrupted versions of the reference text, as transcribed by the ASR system. Corrupted transcription might suffer from Substitution, Deletion, and Insertion, e.g.:

```
Reference: "The cat is sleeping on the mat"
Substitution: "The cat is peeing on the mat"
Deletion: "The cat sleeping on mat"
Insertion: "The cat is always sleeping on
            the mat"
```

Therefore, in real-life scenarios with no Ground Truth, is often impossible to determine what is the right correction. For example, a possible correction for the Substitution example might be: "The white cat is sleeping on the mat", which is also contextually correct. With that being acknowledged, in the context of model evaluation, we possess a clear knowledge of the audio content, allowing us to accurately gauge the proposed correction's alignment with the GT. We will use the following metrics for our experiments:

5.5.1 Word Error Ratio

The Word Error Rate (WER) is the most popular metric [1] used to quantify the accuracy of Automatic Speech Recognition (ASR) systems by measuring the difference between the suggested transcript sentence and the GT in terms of word substitutions (S), words insertions (I), and words deletions (D), and therefore cover the most common errors in ASR. The lower the WER, the higher the accuracy of the predicted transcript.

$$WER = \frac{S + D + I}{N}$$

Where N is the number of words in the reference's text. We used this metric in our project, as the corrections introduced by our REC-ACE model, can be viewed as orthogonal "new" transcribed sentences, and therefore it is straightforward to make use of WER metric.

5.5.2 Exact Match

The Exact Match (EM) metric, is a binary evaluation measure extensively employed across various natural language processing tasks. It signifies whether the output generated by a system perfectly matches the anticipated correct answer. RED-ACE has tagged each word in their dataset with a label that indicates if the ASR got it right or wrong [2]. We used these labels to calculate the percentage of "rights" of each sentence in the dataset.

As depicted in Figure 2, the ASR system typically accurately transcribes a majority of the words. Consequently, we anticipate that our REC-ACE corrections will primarily target a limited number of words in each sentence. Therefore, we expect to see exact matching often after REC-ACE corrections.

5.5.3 BERTScore

BERTScore [13], introduces an innovative approach to assessing the quality of generated text by leveraging contextual embedding from the BERT model.

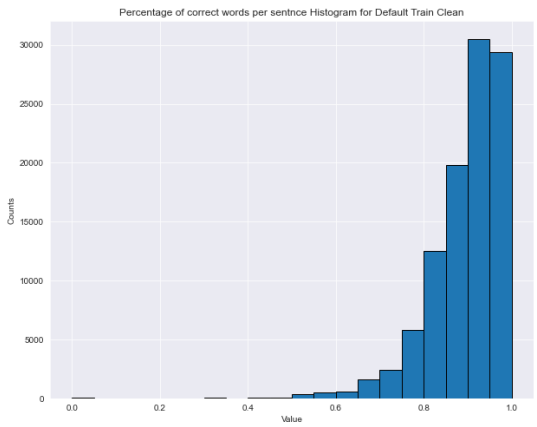

Figure 2: We compute the proportion of accurate ASR transcriptions within each ASR output, as annotated by RED-ACE. The resulting histogram illustrates the distribution of these calculations across all data points within the "Clean" training dataset.

Traditional metrics often struggle to capture semantic and contextual nuances in the generated text. BERTScore addresses this limitation by utilizing contextualized embedding to measure the similarity between generated text and reference text at the token level. This makes BERTScore correlate better with human judgments.
This approach will yield a more comprehensive assessment of the words proposed by our REC-ACE model. Words like "dog" and "cat," which are counted as substitutions in WER but possess semantic relationships, will be captured by the BERT score, providing a richer understanding of their context. The BERT score provides precision, recall, and F1 metrics. We employed the F1 score to gauge the quality of our proposed correction.

6 Results

We conducted separate training on our REC-ACE model on both the "Clean" and "Other" datasets. Then, we assessed each model's performance within its own dataset distribution and also evaluated it using the distribution of the second dataset, to verify the model is still effective across varying audio recording qualities. Table 1 presents the results for models that were trained on the "Clean" dataset and Table 2 for models that were trained on the "Other".

	Clean → Clean			**Clean → Other**		
	WER	**EM**	**BERT**	**WER**	**EM**	**BERT**
ASR	0.124	0.288	0.913	0.272	0.154	**0.812**
T5	0.106	0.347	**0.919**	0.253	0.151	0.809
T5-Prompt	0.111	0.336	0.915	0.256	0.153	0.808
REC-ACE	**0.105**	**0.355**	0.918	**0.251**	**0.157**	0.810
REC-ACE Δ%	-15.3%	+23.3%	+0.5%	-7.7%	+1.9%	-0.2%

Table 1: Evaluating models trained on "Clean" dataset. **Clean → Other** means training on "Clean" and evaluating on "Other". Δ% represent the improvement from ASR baseline.

When contrasted with the T5 base model, the improvement in WER is marginal. However, the exact match metric displays a modest enhancement. This observation implies that the incorporation of the additional signal helps REC-ACE better identify problematic areas in the sentence and make more accurate corrections that maintain the original intended meaning. The same trend can be observed also in Table 2.

	Other → Other			**Other → Clean**		
	WER	**EM**	**BERT**	**WER**	**EM**	**BERT**
ASR	0.272	0.154	0.812	0.124	0.288	0.913
T5	0.230	0.165	0.821	0.094	0.359	0.9227
REC-ACE	**0.226**	**0.172**	**0.824**	**0.090**	**0.378**	**0.925**
REC-ACE Δ%	-16.9%	+11.7%	+1.48%	-27.4%	+31.2%	+1.3%

Table 2: Evaluating models trained on "Other" dataset. **Other → Clean** means training on "Other" and evaluating on "Clean". Δ% represent the improvement from ASR baseline.

Table 2 highlights another noteworthy observation. The models trained on the "Other" dataset appear to perform better on the "Clean" test set compared to the models trained on the "Clean" dataset. The "Other" dataset originates from lower-quality

audio, resulting in a higher number of errors in the ASR hypothesis transcription. Taking this into account along with the observed results, we can hypothesize that models that been exposed to more error examples in training stage have become more robust. Therefore, we observe the models that were trained on "Other" are more robust.

Table 1 also presents the outcomes for the T5-Prompt baseline, exclusively trained on the clean dataset. Interestingly, this baseline under-performed compared to T5 and REC-ACE. This suggests that the introduction of additional information through our engineered prompt might have led to confusion within the model. This aspect should be re-evaluated in future research, employing a diverse range of suggested prompts.

Lastly, we employed the BERT score to assess the contextual alignment of the corrections performed by our model in comparison to the reference text. While this methodology was intended to test the model's efficacy in leveraging prior knowledge and potentially replace the need for human evaluation, there is a difference in the results, so it isn't possible to make any further assumptions or conclusions based on those results.

The results and evaluation pipeline is outlined in the *eval_pipeline* notebook.

7 Conclusions

We introduced REC-ACE, a technique that builds upon the earlier work of RED-ACE in embedding Confidence Scores for ASR Error Correction. REC-ACE is a Transformer-based model, specifically a customized T5 model, where its encoder has been enhanced with an additional embedding layer. This modification allows it to effectively encode both the confidence scores and the transcription hypothesis into a contextualized representation. Through our experiments, we illustrate that combining ASR word-level Confidence Scores with the transcription hypothesis text yields improved outcomes in the Error Correction domain, compared to a T5 model solely trained on the transcription hypothesis. This finding provides support to the conclusions of RED-ACE, indicating that the CS signal indeed offers advantages and that the introduction of the new Embedding layer is an effective means of integrating auxiliary information into the model.

7.1 Examples

Here, we present a few instances that were sampled based on cases where the WER for the REC-ACE hypothesis is reduced by at least 0.4 compared to the ASR hypothesis.

Reference	the feverish colour came into her cheek and the feverish flame into her eye
ASR	favorite color came into a cheeks and the feverish flame into a r i
REC-ACE	the feverish colour came into her cheeks the feverish flame into her eye

Table 3: Example 1

Reference	and thus they journeyed onwards a long long way
ASR	unless they journeyed on words a long long way
REC-ACE	and thus they journeyed onwards a long long way

Table 4: Example 2

Reference	archy checked himself and the boy laughed
ASR	aren't you checked himself in the boy left
REC-ACE	i checked himself and the boy laughed

Table 5: Example 3

More examples, can be found in *eval_pipeline* notebook in the code.

7.2 Future Work

In our future work, we would like to rerun our experiments using a larger-scale T5 architecture. We should assess our model on more datasets to be able to make comparisons with the work of others in the ASR Error Correction field. Furthermore, it could be more aligned with the nature of T5 to rephrase the input prompt as an instruction (e.g., "Correct this: sentence"), considering T5's pretraining methodology. Another avenue we intend to explore involves Prompt Engineering. We are interested in investigating the potential of employing Large Language Models (LLMs) for error correction, using their prior knowledge of the world, as proposed by Ma et al. [8]. This entails providing the supplementary CS vector through prompt engineering and assessing its performance using zero-shot, one-shot, and few-shot learning approaches.

References

[1] Rahhal Errattahi, Asmaa El Hannani, and Hassan Ouahmane. "Automatic speech recognition errors detection and correction: A review". In: *Procedia Computer Science* 128 (2018), pp. 32–37.

[2] Zorik Gekhman et al. "Red-ace: Robust error detection for asr using confidence embeddings". In: *arXiv preprint arXiv:2203.07172* (2022).

[3] Hui Jiang. "Confidence measures for speech recognition: A survey". In: *Speech communication* 45.4 (2005), pp. 455–470.

[4] Colin Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5485–5551.

[5] Youssef Bassil and Paul Semaan. "ASR Context-Sensitive Error Correction Based on Microsoft N-Gram Dataset". In: *CoRR* abs/1203.5262 (2012). arXiv: 1203.5262. URL: http://arxiv.org/abs/1203.5262.

[6] Samrat Dutta et al. "Error Correction in ASR using Sequence-to-Sequence Models". In: *CoRR* abs/2202.01157 (2022). arXiv: 2202.01157. URL: https://arxiv.org/abs/2202.01157.

[7] Shiliang Zhang, Ming Lei, and Zhijie Yan. "Automatic spelling correction with transformer for ctc-based end-to-end speech recognition". In: *arXiv preprint arXiv:1904.10045* (2019).

[8] Rao Ma et al. "Can Generative Large Language Models Perform ASR Error Correction?" In: *arXiv preprint arXiv:2307.04172* (2023).

[9] Rao Ma et al. "N-best T5: Robust ASR Error Correction using Multiple Input Hypotheses and Constrained Decoding Space". In: *arXiv preprint arXiv:2303.00456* (2023).

[10] Haoyu Wang et al. "ASR error correction with augmented transformer for entity retrieval". In: *Interspeech 2020*. 2020. URL: https://www.amazon.science/publications/asr-error-correction-with-augmented-transformer-for-entity-retrieval.

[11] Vassil Panayotov et al. "Librispeech: An ASR corpus based on public domain audio books". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 5206–5210. DOI: 10.1109/ICASSP.2015.7178964.

[12] Taku Kudo and John Richardson. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing*. 2018. arXiv: 1808.06226 [cs.CL].

[13] Tianyi Zhang et al. *BERTScore: Evaluating Text Generation with BERT*. 2020. arXiv: 1904.09675 [cs.CL].