CIS 522 – Final Project – Technical Report

~~Transkripshunists~~ Transcriptionists

April 2022

Team Members:

- Manni Arora; manni; Email: `manni@seas.upenn.edu`
- Pooja Dattatri; poojadat; Email: `poojadat@seas.upenn.edu`
- Shreyas Ramesh; shreyasr; Email: `shreyasr@seas.upenn.edu`

Abstract

We present a comparative study for solving the problem of ASR error correction. We treat ASR error correction as a post processing task. We have experimented with grammar correction models and GPT-3 to solve this task. We have observed that grammar correction models can help correct contextual errors and if fine-tuned on large volume of data can potentially be very useful for solving the problem of ASR error correction.

1 Introduction

Pre-trained off-the-shelf Automatic Speech Recognition (ASR) systems are becoming a more feasible option for firms of all sizes developing speech-based products. Despite the fact that these ASR systems are trained on enormous volumes of data, domain mismatch remains an issue for many customers that want to use the service as-is, resulting in less-than-optimal performance. We present a comparative study for solving the problem of ASR error correction. We take two high level approaches to solve the task of error correction. The first approach we use is to finetune pre-trained ASR models. In the second approach, we treat ASR error correction as a post processing task and finetune models to correct the errors in the ASR output. We experiment with grammar correction models and GPT-3 to solve this task.

2 Related Work

Previous work on ASR error correction or end-to-end ASR models was explored using multiple techniques. Youssef Basil et al, [2] explored the problem of spelling correction and proposed an error correction algorithm using Microsoft's N-gram dataset and focused on post-editing methods. Hoang Long Nguyen et al, [5] had explored this problem by proposing a rewritten query that is generated by incorporating the initial incorrect utterance and the follow-up correction utterance, which is then used to re-estimate the intent behind the utterance and correct it. Wang, Haoyu et al., [7] proposed a transformer model that encodes the phonetic information of the entity and conducts sequential attention on the text embedding as well as the phoneme sequence. This works well for domain specific datasets and vocabulary.

3 Dataset and Features

AMI Meeting Corpus: There are 100 hours of meeting recordings in the AMI Meeting Corpus. A variety of signals are used in the recordings, all of which are synchronized to a similar timeline. Close-talking and far-field microphones, individual and room-view video cameras, and slide projector and electronic whiteboard output are among them. During the meetings, the attendees have access to unsynchronized pens that record what is written. The discussions were recorded in English in three distinct rooms with varying acoustic features, with the majority of the participants being non-native English speakers. We use 9 hours of meeting recordings from this corpus for our experiments. Each recording has corresponding transcripts in English. We use these transcripts as gold labels for the error correction task.

4 Methodology

We use pre-trained ASR models to generate transcripts for audio files. We do not have a non-DL benchmark but instead treat the generation of transcripts from a pre-trained ASR model as our baseline. We then treat ASR error correction as a post processing task and finetune models to correct the errors in the ASR output. We experiment with grammar correction models and GPT-3 to solve this task.

4.1 Baseline

We treat the results from a pre-trained ASR model as our baseline results. We have used Facebook's Wav2letter as our pre-trained ASR model. ASR models output transcripts for given audio files. Wav2letter implements the architecture proposed in Ronan et al. [4]. It provides pre-trained models trained on the Librispeech dataset. We fed this model audio files from the AMI Meeting Corpus to obtain the corresponding transcripts.
We would also like to take this space to highlight a few challenges we faced while obtaining the transcripts. Our first task was to choose a pre-trained ASR model. We had make sure that the model was open-source and that the training data used in the pre-trained model was not the same as our evaluation data set.

The second thing that we had to deal with was the length of our audio files. Wav2letter does not take audio files larger than 10 seconds long. So, we had to split our audio files into chunks before feeding it to the pre-trained model. Another challenge we had was to ensure that the type of the audio files was *.wav* and that the sample-rate was 16000. Since Wav2letter was trained on these type of audio files, the model would best work if the original audio was recorded with sample-rate 16000 and stored as *.wav*. Even though it is possible to convert any other audio file to this format, we observed that quality is lost in the conversion and that the model does not perform well.

4.2 Text-to-Text Transformer

To improve our baseline results, we experimented with fine-tuning two architectures. The first was to fine-tune a text-to-text transformer architecture. We found the problem of grammar correction to be the most similar to our proposed problem of ASR error correction. We used a grammar corrector model from HuggingFace[1]. This model is built on Goggle's T5 [6], used for all NLP tasks in a text-to-text-format where the input and output are always text strings, in contrast to BERT-style models that can only output either a class label or a span of the input. Transcripts from the pretrained-ASR model along with the original transcripts downloaded with the dataset of AMI audio files were used for fine-tuning this model. We had a separate test set on which we evaluated the finetuned model. One drawback of this model is that it works at sentence levels and has been trained on 64 length sentences, so not yet suitable for long prose or paragraphs. So, our evaluation of this model was also on sentenced truncated to lengths of 64.

4.3 OpenAI's GPT-3

Generative Pre-trained Transformer 3(GPT-3) is an autoregressive language model that uses deep learning to produce human-like text. It is the third-generation language prediction model in the GPT-n series created by OpenAI [3]. . GPT-3 is the incredibly popular current state-of-the-art Language model, which can even generate code. We tried zero-shot learning, few-shot learning and fine-tuning GPT-3. We created an OpenAI account, and used the openAI api to finetune and evaluate the model. Similar to the previous model, transcripts from the pretrained-ASR model along with the original transcripts downloaded with the dataset of AMI audio files were used for fine-tuning this model. A sample of the prompt and completion given to GPT-3 is as shown in 1. We had a separate test set on which we evaluated the fine-tuned model. We compare the results of all these models in the next section.

4.4 Hyper Parameter Optimization

We tweaked some of the hyperparameters while fine-tuning the Grammar corrector Text-to-Text Transformer: *learning_rate, weight_decay, adam_beta1, adam_beta2,*

```
Prompt =
{
    "ASR output": "okay preject finance setling price twenty
     ↪  five euros profet am fifty million eroes"
}
Completion =
{
    "ASR correction": "okay project finance selling price
     ↪  twenty five euros profit aim fifty million euros"
}
```

Listing 1: An example of the prompt and completion given to fine-tune GPT-3

num_train_epochs. This model uses Adam as the optimizer. The descriptions of each of these hyper parameters and the values that we used in our model is given below:

- *learning_rate* specifies the initial learning rate for Adam. We used a value of 5e-5
- *weight_decay* specifies the weight decay to apply (if not zero). We left it at default to 0
- *adam_beta1* specifies the beta1 hyperparameter for the Adam optimizer.We used a value of 0.9
- *adam_beta2* specifies the beta2 hyperparameter for the Adam optimizer.We used a value of 0.999
- *num_train_epochs* specifies the total number of training epochs to perform. We used a value of 7

5 Results

Correcting the errors that an ASR system makes as a post-processing task improves the transcriptions by a significant amount. This can be seen in the reduction of Word Error Rate (WER) scores after fine-tuning the model to correct errors as shown in 1

Model	**Word Error Rate (WER)**
Wav2Letter (Baseline)	26.6%
Text-to-Text Transformer	25.36%
GPT-3 fine-tuned	27.14%

Table 1: WER comparison

WER is the best metric that we could think of, useful for our use case. WER is calculated as shown in the 1.

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

where

- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- C is the number of correct words,
- N is the number of words in the reference (N=S+D+C)

Figure 1: WER Formula

Here, substitutions, deletions and insertions are done in the transcripts produced by the ASR system to correct it to the gold transcripts of the dataset.

We can see that the WER score has reduced after fine-tuning a grammar correction Text-to-Text Transformer model with our data. The decrease in the training loss while fine-tuning this model is shown in 2 and the trend in the validation loss is shown in 3. Some of the results of this model are shown in 4.

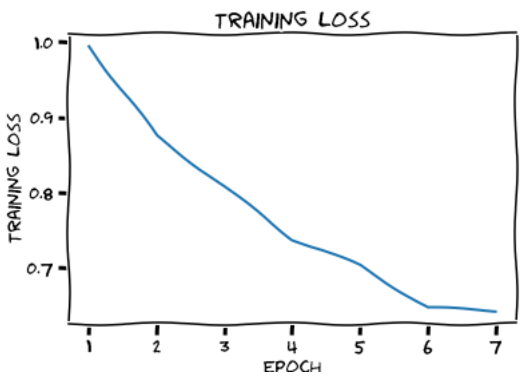


Figure 2: Training Loss

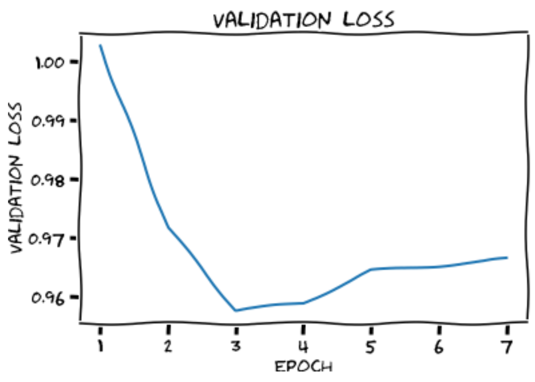
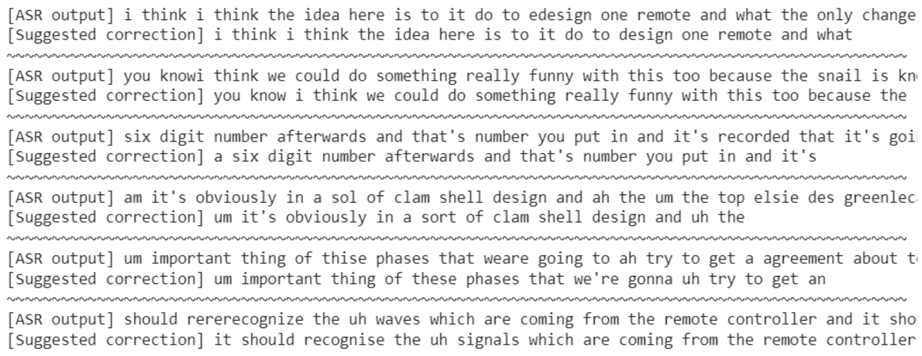


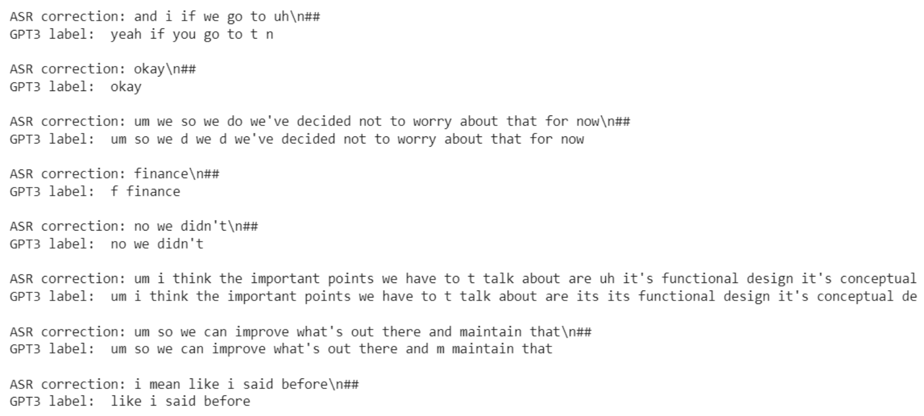
Figure 3: Validation Loss



```
[ASR output] i think i think the idea here is to it do to edesign one remote and what the only change
[Suggested correction] i think i think the idea here is to it do to design one remote and what
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[ASR output] you knowi think we could do something really funny with this too because the snail is kn
[Suggested correction] you know i think we could do something really funny with this too because the
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[ASR output] six digit number afterwards and that's number you put in and it's recorded that it's goi
[Suggested correction] a six digit number afterwards and that's number you put in and it's
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[ASR output] am it's obviously in a sol of clam shell design and ah the um the top elsie des greenlec
[Suggested correction] um it's obviously in a sort of clam shell design and uh the
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[ASR output] um important thing of thise phases that weare going to ah try to get a agreement about t
[Suggested correction] um important thing of these phases that we're gonna uh try to get an
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[ASR output] should rererecognize the uh waves which are coming from the remote controller and it sho
[Suggested correction] it should recognise the uh signals which are coming from the remote controller
```

Figure 4: Text-to-Text Transformer Results

We observe that the WER score for the state-of-the-art GPT-3 model has increased from the baseline. Some of the results from GPT-3 are shown in 5.



```
ASR correction: and i if we go to uh\n##
GPT3 label:  yeah if you go to t n

ASR correction: okay\n##
GPT3 label:  okay

ASR correction: um we so we do we've decided not to worry about that for now\n##
GPT3 label:  um so we d we d we've decided not to worry about that for now

ASR correction: finance\n##
GPT3 label:  f finance

ASR correction: no we didn't\n##
GPT3 label:  no we didn't

ASR correction: um i think the important points we have to t talk about are uh it's functional design it's conceptual
GPT3 label:  um i think the important points we have to t talk about are its its functional design it's conceptual de

ASR correction: um so we can improve what's out there and maintain that\n##
GPT3 label:  um so we can improve what's out there and m maintain that

ASR correction: i mean like i said before\n##
GPT3 label:  like i said before
```

Figure 5: GPT-3 Results

6 Discussion

ASR systems are prone to make errors in the transcripts that they produce. A lot of work has been done on improving the ASR system itself but very few people have worked on ASR error correction as a post-processing task. Even these work do not include additional context that might help correct these errors or have domain adaptation capabilities. In our project, we have attempted to correct ASR errors as a post-processing task, after narrowing the domain of audio recordings to meeting recordings.

6.1 Findings

We use the WER score from off-the-shelf ASR system (Wav2Letter) as baseline. We can see that the WER score has reduced after fine-tuning a grammar correction model with our data. With more compute power and more training data, we believe that this score could be reduced further. In contrast, we observe that the WER score for the state-of-the-art GPT-3 model has increased from the baseline. We think that this is because GPT-3 is trained to perform such a wide variety of tasks that it might not be specific enough for our use case. Fine-tuning GPT-3 with the little amount of data that we did would be like scratching the surface because GPT-3 is trained on such a large amount of data. We also used the version of GPT-3 with the least cost - babbage. Fine-tuning costlier and bigger models might have resulted in a better WER score.

Current state-of-the-art WER score for ASR systems is 19% but this number is before performing the post-processing task. We believe that our proposed method, when run on machines with good compute power will beat the current state-of-the-art WER score.

As a social impact, ASR systems could be used on voices with speech impairments and post-processing ASR errors using context could help understand them better.

6.2 Limitations and Ethical Considerations

There are three major potential limitations with our approach. Firstly, the model depends on the qualty of the raw data and its specifications. Testing a model trained on audio files of a particular sample rate using data of a different sample rates would lead to massive errors. Up-/downsampling the files to suit the requirements generated unnecessary noise to our training, lowering its performance.

Secondly, although there has been improvement in reducing the WER score when fine-tuning GPT-3, this has its own limitations. As we fine-tuned the model to perform well in our dataset, it may not be accurate enough when testing with a completely different dataset.

Lastly, running the entire 100 hours AMI Meeting corpus over the model is computationally expensive. Hence we were able to train the model using 9 hours of the meeting recordings from the corpus. Utilizing a lager dataset would help improve the performance of the model and can generalize better with lower error rates.

6.3 Future Research Directions

This project demonstrates how fine-tuning GPT-3 improve ASR Correction and shows how it could be a good post-processing tool for existing ASR systems. There is tremendous scope in this topic. As future work, we could train the model to work well with different recording codecs and sample rates. Another major potential would be extending this for multiple languages which would improve translation as well as accurate ASR. We could work towards reducing the speed of computation for real time transcription, with minimal to low errors.

7 Conclusions

We demonstrate effectiveness of grammar correction and GPT-3 in ASR error correction using fine-tuning to help derive WER reductions.This study demonstrates that existing ASR systems can be adapted for specific datasets by post-processing the ASR hypothesis with finetuning models like GPT-3. These methods are easy to implement and can improve ASR quality for domain-specific applications. The study evaluates the ASR quality improvements in terms of WER. The results show that the ASR transcription quality can be improved with these methods.

References

[1] URL: `https://huggingface.co/prithivida/grammar_error_correcter_v1`.

[2] Youssef Bassil and Paul Semaan. "ASR Context-Sensitive Error Correction Based on Microsoft N-Gram Dataset". In: (2012). DOI: `10.48550/ARXIV.1203.5262`. URL: `https://arxiv.org/abs/1203.5262`.

[3] Tom B. Brown et al. *Language Models are Few-Shot Learners.* 2020. DOI: `10.48550/ARXIV.2005.14165`. URL: `https://arxiv.org/abs/2005.14165`.

[4] Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. *Wav2Letter: an End-to-End ConvNet-based Speech Recognition System.* 2016. DOI: `10.48550/ARXIV.1609.03193`. URL: `https://arxiv.org/abs/1609.03193`.

[5] Hoang Long Nguyen et al. *User-Initiated Repetition-Based Recovery in Multi-Utterance Dialogue Systems.* 2021. DOI: `10.48550/ARXIV.2108.01208`. URL: `https://arxiv.org/abs/2108.01208`.

[6] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.* 2019. DOI: `10.48550/ARXIV.1910.10683`. URL: `https://arxiv.org/abs/1910.10683`.

[7] Haoyu Wang et al. "ASR Error Correction with Augmented Transformer for Entity Retrieval". In: *INTERSPEECH.* 2020.