ASR Error Correction with Augmented Transformer for Entity Retrieval

Haoyu Wang†, Shuyan Dong†, Yue Liu, James Logan, Ashish Kumar Agrawal, Yang Liu

Amazon Alexa, U.S.A

† Both authors contributed equally to this work

{wanhaoyu, shuyand, lyu, jimlog, ashiagra, yangliud}@amazon.com

Abstract

Domain-agnostic Automatic Speech Recognition (ASR) systems suffer from the issue of mistranscribing domain-specific words, which leads to failures in downstream tasks. In this paper, we present a post-editing ASR error correction method using the Transformer model for entity mention correction and retrieval. Specifically, we propose a novel augmented variant of the Transformer model that encodes both the word and phoneme sequence of an entity, and attends to phoneme information in addition to word-level information during decoding to correct mistranscribed named entities. We evaluate our method on both the ASR error correction task and the downstream retrieval task. Our method achieves 48.08% entity error rate (EER) reduction in ASR error correction task and 26.74% mean reciprocal rank (MRR) improvement for the retrieval task. In addition, our augmented Transformer model significantly outperforms the vanilla Transformer model with 17.89% EER reduction and 1.98% MRR increase, demonstrating the effectiveness of incorporating phoneme information in the correction model.

Index Terms: speech recognition, error correction, entity retrieval, transformer

1. Introduction

In spite of the recent advancement in general-purpose ASR systems [1], recognizing domain-specific words remains a big challenge. Many of the words involve rare, non-idiomatic, or code-switched proper nouns and acronyms that are out-of-vocabulary of general-purpose ASR models. As a result, they are often mistranscribed to common words in the vocabulary. In a task-oriented spoken dialog system, downstream tasks such as Natural Language Understanding (NLU) and Entity Retrieval (ER) rely on ASR transcripts, and ASR errors will propagate to downstream stages, leading to unfulfilled user requests.

One solution to address this problem is using a domain-specific ASR system [2]. However, such approach is usually data hungry and computationally expensive. Another solution is to enrich the ASR systems with more words and phrases with a customized language model [3], but how to balance words from each domain arises as a new challenge. Moreover, most ASR systems will not scale well when each customized list of domain-specific phrases contains hundreds of thousands of entities in a huge catalog.

In a spoken dialogue system, NLU takes the ASR transcriptions and performs domain/intent classification and Named Entity Recognition (NER). NER labels the entity mentions with entity types. An Entity Retrieval (ER) component then takes an entity mention as a query and searches the catalog based on the intent and entity type to retrieve the most relevant real world object for the entity mention. Consider a speech-based virtual assistant that helps users explore places and local businesses, for an utterance *how long does it take to go to enloe hospital?*, NER labels *enloe hospital* as *PlaceName*, and then ER takes this as a query and links to an entity in the catalog inventory, returning the most relevant place to the user. Since "enloe" is not a common word, "enloe hospital" may get mistranscribed as "in lo hospital", and the ER system in turn may end up retrieving nothing or an irrelevant place from its catalog, which adversely affects customer experience. Therefore, being able to handle such ASR errors before retrieval is important for ER performance.

In this work, we tackle this challenge by proposing a domain specific post-editing model to correct ASR errors on entity mentions before ER. While existing work generally uses a text-based sequence to sequence model to correct such errors, we propose a novel neural correction module that augments the vanilla Transformer model [4] by deeply infusing and attending phonetic information of the entity mention. We demonstrate significant improvement on two widely used domains for a voice assistant system including *shopping* and *local search*. Our main contribution includes two folds: (1) we propose a novel augmented generalized variant of the Transformer model that allows encoding different types or lengths of the input signals with the flexibility to control and balance the weight of each. (2) we develop a domain-specific post-editing error correction system that leverages entity catalogs based on the augmented Transformer with text and phoneme as input sequences, and show that our system is able to effectively correct entity errors and significantly improve ER performance.

2. Related Work

Previous work has explored error-corrective language modeling or end-to-end ASR using various techniques [3, 5, 6, 7]. Others focused on post-editing methods that target spelling errors introduced by ASR [8, 9]. Reranking ASR n-best hypotheses is also a popular topic for error recovery. In-domain and contextual information has shown to be effective in reranking ASR n-best hypotheses [10, 11]. There are also joint learning efforts [12, 13, 14] that show promises in incorporating upstream signals for downstream uses. To our knowledge, there are very few correction methods that focus on entity retrieval as a target task. [15] is highly relevant to our use case. It uses text and phoneme similarities to make retrieval robust to ASR errors for person names in their entity retrieval model.

Another relevant field is noisy natural language correction. Some work in this field focuses on grammatical errors such as article, verb tense and preposition [16]. Recent work [17] shows promising results on tackling orthographic errors that deal with spelling, non-idiomatic expression, or punctuation. However, most of these approaches do not impact the retrieval results for named entities because search engines are generally mature enough to support fuzzy queries to handle minor errors including spelling and grammatical errors.

3. Proposed Method

Our proposed augmented generalized transformer solution for entity mention correction builds upon and extends the Transformer model [4] with a sequential attention mechanism to infuse extra signal. Without loss of generality, we use phonetic information as an example, which ties to our application of ASR error correction. The model architecture is illustrated in Figure 1. It is worth noting that our solution is applicable to other transformer-based encoder-decoder architectures, with other types of signals, and to other downstream tasks.

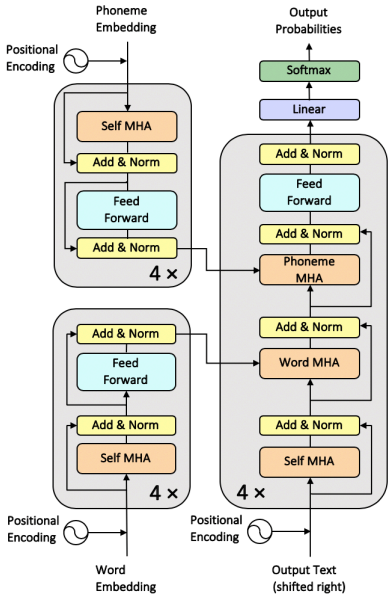

Figure 1: *Augmented Transformer using different types of information with sequential attention in the decoder. Comparing to the vanilla Transformer, the model conducts two attentions sequentially on the text embedding and the phoneme embedding. Three variants of the decoder architecture are introduced in Section 3.2.*

The input of the correction model is an entity mention defined as a sequence of words $\boldsymbol{w} = \{w_1, w_2, ...w_N\}$ tokenized by sentencepiece tokenization [18] and a sequence of phonemes $\boldsymbol{p} = \{p_1, p_2, ...p_M\}$ representing the entity's pronunciation. The output of the model is expected to be a rewritten entity represented by a new sequence of words $\hat{\boldsymbol{w}} = \{\hat{w}_1, \hat{w}_2, ...\hat{w}_{N'}\}$.

3.1. Encoder

The Transformer [4] model has become the new state-of-the-art model in recent years and has been successfully adopted by many NLP tasks [19]. In this work, we extend the model by providing a second view based on the phonetic information. The intuition is that ASR systems typically mistranscribe entities to acoustically similar words. We use a stacked Transformer encoder to encode the text sequence $\boldsymbol{w}$ into a vector space $H^w = \{h_1^w, h_2^w, ...h_N^w\}$, and similarly another stacked Transformer encoder to encode phoneme sequence $\boldsymbol{p}$ as $H^p = \{h_1^p, h_2^p, ...h_M^p\}$.

3.2. Decoder

Recall that there are two attention mechanisms in the Transformer decoder: self-attention and encoder-decoder attention. We keep the self-attention part the same as the vanilla Transformer, and propose three approaches to use information from the two sequences for the encoder-decoder attention.

• **Sequential Attention**: We conduct the encoder-decoder attention twice in sequence to infuse information from the two input signals. In our approach, at each Transformer layer of the decoder, we first compute the attention output A^w between the text encoder's output H^w and decoder's self-attention output $H^{\hat{w}}$ by:

$$A^w = softmax(\frac{(Q^w H^w)(K^w H^w)^T}{\sqrt{d_k}})(V^w H^{\hat{w}}) \quad (1)$$

where $Q^w, K^w, V^w \in \mathbb{R}^{d_k \times d_k}$ are the parameters of the model, and d_k is the dimension of the output from the self-attention layer.

Similarly, we take the phoneme encoder's output H^p and compute the attention with previous output A^w:

$$A^p = softmax(\frac{(Q^p H^p)(K^p H^p)^T}{\sqrt{d_k}})(V^p A^w) \quad (2)$$

where $Q^p, K^p, V^p \in \mathbb{R}^{d_k \times d_k}$ are the parameters of the model. In this approach, the second attention using the phoneme encoder's output could also be treated as a phonetic regularization on the first attention output (word-based).

• **Concatenated Attention**: Different from the above sequential approach, here we compute the second attention between the phoneme encoder's output H^p and the decoder's self-attention output $H^{\hat{w}}$ by:

$$\tilde{A}^p = softmax(\frac{(Q^p H^p)(K^p H^p)^T}{\sqrt{d_k}})(V^p H^{\hat{w}}) \quad (3)$$

Then we concatenate A^w and $\tilde{A}^p$, followed by a dense layer to project it back to the original model size d_k.

• **Pooled Attention**: On top of the concatenated attention approach, we apply a max-pooling between A^w and $\tilde{A}^p$ to merge the information from the two sources into a representation of size d_k.

All of these three approaches generate an encoder-decoder attention representation. Same as the vanilla Transformer, the attention output will be followed by a fully-connected feed-forward layer with layer normalization to generate the output of the current decoder layer. The output will be used either as input to the next layer in the stacked decoder or as the representation to predict the output. Note that because the lengths of the input sequences do not always match ($N \neq M$), we cannot perform a positional level representation concatenation of the outputs from the two encoders. In the future, we also plan to explore concatenating the two encoded sequences, e.g., resulting in a sequence of length of N+M.

3.3. Training

Except for the changes on the attention mechanism for the decoder, we follow the same setup of the vanilla Transformer including model architecture, positional encoding, and learning rate. For our error correction task, we introduce two simple changes in model training to (1) encourage the model to leverage phonetic information and (2) prevent the model from over correcting error-free entity mentions.

• **Text Input Dropout**: In order to encourage the model to leverage phoneme information to conduct error correction rather than merely memorizing all the error patterns seen in the training data, we apply a simple yet efficient technique:

dropout. The goal is to randomly drop out text input with a rate of α to ensure the model will learn from the phoneme input. The text input will be masked as $\boldsymbol{w} = \{mask_1, mask_2, ...mask_N\}$ when an example is randomly picked to apply dropout to.

• **Clean Input**: In a real-world production system, the model does not know whether the incoming ASR transcribed entity contains ASR error or not, hence it is important that the model does not over-correct error-free entity input. We guide the model to learn this by randomly swapping a mistranscribed entity with an error-free one with a probability of β.

4. Dataset Creation

4.1. Data Sources

We use public data from two domains to create benchmark datasets to evaluate our model. We extract "product_title" from Amazon Review Dataset [20] to create a shopping product catalog and use "name" from Yelp Open Dataset [21] to create a local search catalog. This gives 15,651,951 unique shopping entities and 133,637 unique local search entities. We de-noise the catalog entries by keeping the named entities with fewer than 10 tokens, and randomly sample 100,000 unique instances from each catalog to form a dataset for each domain.

4.2. Error Simulator

To ingest ASR errors into the dataset, we adopt a text-level error simulator [22]. It is based on an n-gram confusion matrix learned from hypothesis-reference pairs from a real dialog system. One common drawback for such simulated data is that the model might easily learn the reverse of the confusion matrix by memorizing all error patterns, and cannot generalize well to new error patterns. In order to avoid such an effect, we partition the error patterns randomly into two partitions, and use one for training and the other for testing. This assures that all the error patterns in the test data are unseen in training data. For experimental purpose, we simulate the errors in the datasets with the following two approaches.

4.2.1. SIM1: Corpus Level Error Simulation

In this approach, we apply the error simulator to introduce errors with a target WER of 8%-10% to create the test set. The WER is similar to what we observe from commercial dialogue systems, and thus we expect the experimental results on this data to reflect the actual gain/loss on real use cases. The resulting WER on the test set is higher than the target WER because the token length of our entity names is short. As for training, in order to generate more data, we sample error 10 times for each entity and collect all the unique error cases. This results in higher WER than that of the test set, as well as higher Entity Error Rate (EER), defined as the percentage of the entities having ASR errors. The summary statistics for training and test data are shown in Table 1. In Table 2 we provide examples of the simulation results.

	Yelp business		Amazon product	
	training	test	training	test
# Instances	442,067	100,000	771,994	100,000
Avg length	3.65	3.24	6.61	6.01
WER	28.46%	15.43%	21.41%	12.67%
EER	79.05%	38.50%	88.83%	51.97%

Table 1: *Summary statistics of SIM1.*

	Entity Text with ASR Error	Ground Truth
Yelp	rv penis pooper roofing	rv phoenix cooper roofing
Amazon	speed soothing love right	speed sewing love riot

Table 2: *Examples from the simulation results.*

4.2.2. SIM2: Entity Level Error Simulation

Although the aforementioned error simulation approach tries to loop through each entity and inject errors based on the target WER, one entity might have no error and another might have multiple errors so there is no guarantee that such probabilistic approach will give the same WER for each entity. As a result, a substantial amount of the entities do not have any simulated error using the above method SIM1 (61.5% Yelp and 48.03% Amazon), which prevents us from obtaining performance on those entities. To solve this issue, we construct another dataset by injecting exactly one error on one token probabilistically for each entity. We randomly select a token from the named entity and replace the token by an error token that is sampled probabilistically from the confusion matrix. If the candidate error list is empty, we randomly select another token from the named entity and repeat this process. In the case that we exhaust all the tokens of a named entity and all the candidate error lists are empty, we exclude this entity from the test set (1.8% for shopping and 5.4% for local search) for reporting purpose. Similar to the previous approach to create the training data, in order to have more error variations for the model to learn and generalize better, we probabilistically sample 5 errors (at most) for each token within an entity, which yields a maximum of $5 \times$ (# tokens) examples. Table 3 shows the statistics of this dataset.

	Yelp business		Amazon product	
	training	test	training	test
# Instances	718,702	94,598	896,797	98,151
Avg length	3.63	3.35	6.28	6.08
WER	28.29%	30.73%	16.32%	16.78%
EER	100%	100%	100%	100%

Table 3: *Summary statistics of SIM2.*

5. Experiments

For all the experiments, we employ the same set of model hyper-parameters. Both the encoder and decoder are composed of 4 identical layers of transformers with a hidden size of $d_k = 128$, and 8 attention heads. The size of the feed-forward layer is 512. We apply the same optimizer as the original transformer with a warm-up steps of 4,000. The batch size is 512 and we train each model for 50 epochs. We did not conduct any experiment to tune those hyper-parameters as that is not the main focus of the work. Phoneme sequences are generated using CMU G2P model [23] for each token within the dataset.

5.1. Text Rewrite and Entity Retrieval

In order to evaluate the performance of the rewrite, we calculate two metrics: word error rate (WER) and entity error rate (EER). WER is token-level error rate derived from the Levenshtein distance and reflects the token level performance. EER is the percentage of incorrect entities and is an entity level metric, which is more crucial to the downstream ER performance.

	yelp business				amazon product			
	WER	EER	Prec@1	MRR	WER	EER	Prec@1	MRR
Original	15.43	38.50	80.29	82.69	12.73	51.97	94.46	95.00
Vanilla Trans	14.46	32.58	85.12	86.94	11.01	41.93	95.16	95.18
Augmented Trans	**12.91**	**29.50**	**86.77**	**88.29**	**9.97**	**36.06**	**95.76**	**96.29**

Table 4: *Experiment results on SIM1, which uses the corpus level error simulation method described in 4.2.1.*

	yelp business				amazon product			
	WER	EER	Prec@1	MRR	WER	EER	Prec@1	MRR
Original	30.73	100.00	57.47	63.36	16.78	100.00	90.86	92.26
Vanilla Trans	23.09	63.23	74.22	78.74	10.86	58.54	95.35	96.10
Augmented Trans	**19.60**	**51.92**	**76.73**	**80.30**	**10.55**	**53.15**	**95.54**	**96.14**

Table 5: *Experiment results on SIM2, which uses the entity level error simulation method described in 4.2.2.*

We also measure the impact of the model on the target entity retrieval task. As a popular approach adapted by many retrieval systems, an Elastic search engine is set up and all the entities within the catalog are ingested. We compare the retrieval performance when using the original entity text versus the rewritten entity text. We use Precision at 1 (Prec@1) and Mean Reciprocal Rank (MRR) as the evaluation metrics for our retrieval task.

The experiment results in Table 4 and 5 demonstrate that our approach of adding phonetic information yields a significant WER and EER reduction for both data sets. The SIM2 data set has a relatively larger improvement compared to the SIM1 data set because in SIM2 all the entities have errors. Comparing the improvement on Yelp business and Amazon product, we observe that the phonetic information plays a more important role in error correction on relative shorter text. When there are fewer tokens in the entity, an error is more fatal to the final ER result as it could lead to a totally different retrieval result. For the relatively longer entities in Amazon product, the search engine itself may still be robust by relying on other words to conduct the search correctly.

5.2. Detailed Analysis

First, we compare the setup in the above experiments with two alternative decoder architectures described in Section 3.2 on the SIM2 data set. From the results in Table 6, we can clearly see the benefit of phonetic information used in our augmented Transformer architectures. In addition, the sequential attention approach achieves the best performance on both domains, outperforming the other two methods of combining the word and phonetic information in the decoder.

	yelp business		amazon product	
	WER	ERR	WER	ERR
text only	23.09	63.23	10.86	58.54
concatenated	20.37	54.60	10.90	54.86
pooled	20.35	53.91	10.80	54.25
sequential	**19.60**	**51.92**	**10.55**	**53.15**

Table 6: *Performance of different model architectures.*

Next, we discuss the effect of hyper-parameter α and β through experiments on SIM2 data set. From the results in Figure 2, we notice a substantial improvement by introducing the text dropout hyper-parameter α since we encourage the model to rely on the phonetic information for better generalization capability. However, increasing the dropout rate will eventually lead to a performance drop as text information is also crucial to the rewrite task. We find an α around 0.5 is efficient to balance the two factors and obtain the optimal performance.

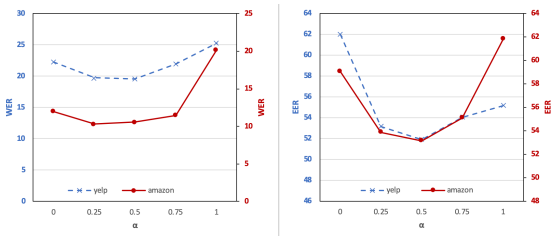

Figure 2: *Model performance using different α.*

We also measure the performance using different clean text rate β. In this experiment, we evaluate the impact of this parameter on two data sets: (1) fully corrupted data and (2) fully clean data (i.e., error-free entities). Ideally we would like the model to conduct the right rewrite for corrupted input while keeping the clean input unchanged. As shown in Figure 3, having a small β will result in a conservative model that has a very low WER and EER on the clean data, but also a large WER and EER on the corrupted data. A large value of β, on the other hand, will make the model to be more aggressive and achieve a better WER and EER on the corrupted data, at a cost of a higher WER and EER on the clean data. We find that a β around 0.5 is reasonable to balance the performance on both data sets, but this parameter can be tuned depending on the actual use case.

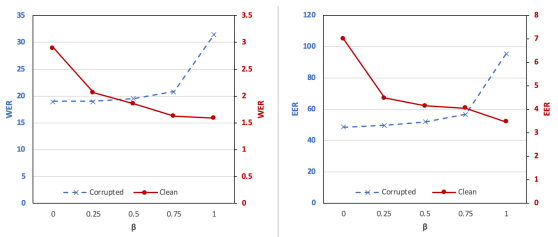

Figure 3: *Model performance using different β.*

6. Conclusion

In summary, we proposed a new solution to infuse additional information into a Transformer-based sequence to sequence model. With the proposed augmented Transformer with phonetic signals, we achieve significant improvement on noisy entity text rewrite, which further improves the downstream entity retrieval task. Our model architecture is flexible and can be used to infuse input of other modalities for broader applications.

7. References

[1] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[2] S. Deena, R. W. Ng, P. Madhyashta, L. Specia, and T. Hain, "Semi-supervised adaptation of rnnlms by fine-tuning with domain-specific auxiliary features," in *Proceedings of INTERSPEECH 2017: Conference of the International Speech Communication Association*. ISCA, 2017, pp. 2715–2719.

[3] J. Guo, T. N. Sainath, and R. J. Weiss, "A spelling correction model for end-to-end speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5651–5655.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[5] B. Roark, M. Saraclar, and M. Collins, "Corrective language modeling for large vocabulary asr with the perceptron algorithm," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 2004, pp. I–749.

[6] T. Tanaka, R. Masumura, H. Masataki, and Y. Aono, "Neural error corrective language models for automatic speech recognition," in *Proc. Interspeech 2018*, 2018, pp. 401–405.

[7] S. Kumar, M. Nirschl, D. Holtmann-Rice, H. Liao, A. T. Suresh, and F. Yu, "Lattice rescoring strategies for long short term memory language models in speech recognition," 2017.

[8] Y. Bassil and P. Semaan, "Asr context-sensitive error correction based on microsoft n-gram dataset," 2012.

[9] E. K. Ringger and J. F. Allen, "Error correction via a post-processor for continuous speech recognition," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1, 1996, pp. 427–430 vol. 1.

[10] R. Corona, J. Thomason, and R. Mooney, "Improving black-box speech recognition using semantic parsing," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2017, pp. 122–127.

[11] R. Jonson, "Dialogue context-based re-ranking of asr hypotheses," in *2006 IEEE Spoken Language Technology Workshop*, 2006, pp. 174–177.

[12] F. Morbini, K. Audhkhasi, R. Artstein, M. Van, S. K. Sagae, P. Georgiou, D. R. Traum, and S. Narayanan, "A reranking approach for recognition and classification of speech input in conversational dialogue systems," 2012.

[13] P. Haghani, A. Narayanan, M. Bacchiani, G. Chuang, N. Gaur, P. J. Moreno, R. Prabhavalkar, Z. Qu, and A. Waters, "From audio to semantics: Approaches to end-to-end spoken language understanding," *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 720–726, 2018.

[14] Y. Weng, S. S. Miryala, C. Khatri, R. Wang, H. Zheng, P. Molino, M. Namazifar, A. Papangelis, H. Williams, F. Bell, and G. Tur, "Joint contextual modeling for asr correction and language understanding," *ArXiv*, vol. abs/2002.00750, 2020.

[15] A. Raghuvanshi, V. Ramakrishnan, V. Embar, L. Carroll, and K. Raghunathan, "Entity resolution for noisy asr transcripts," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, 2019, pp. 61–66.

[16] H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, "The CoNLL-2014 shared task on grammatical error correction," in *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, 2014, pp. 1–14.

[17] Z. Xie, A. Avati, N. Arivazhagan, D. Jurafsky, and A. Y. Ng, "Neural language correction with character-based attention," 2016.

[18] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *arXiv preprint arXiv:1804.10959*, 2018.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[20] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *proceedings of the 25th international conference on world wide web*, 2016, pp. 507–517.

[21] *Yelp Dataset*, https://www.yelp.com/dataset/.

[22] M. Fazel-Zarandi, L. Wang, A. Tiwari, and S. Matsoukas, "Investigation of error simulation techniques for learning dialog policies for conversational error recovery," *ArXiv*, vol. abs/1911.03378, 2019.

[23] *CMU Sphinx*, https://github.com/cmusphinx/g2p-seq2seq.