User-Initiated Repetition-Based Recovery in Multi-Utterance Dialogue Systems

Hoang Long Nguyen, Vincent Renkens, Joris Pelemans, Srividya Pranavi Potharaju,
Anil Kumar Nalamalapu, Murat Akbacak

Apple, 1 Apple Park Way
Cupertino, California, USA

{romanhoangnguyen_long, vrenkens, jpelemans, spotharaju}@apple.com,
{anilkumar_nalamalapu, makbacak}@apple.com

Abstract

Recognition errors are common in human communication. Similar errors often lead to unwanted behaviour in dialogue systems or virtual assistants. In human communication, we can recover from them by repeating misrecognized words or phrases; however in human-machine communication this recovery mechanism is not available. In this paper, we attempt to bridge this gap and present a system that allows a user to correct speech recognition errors in a virtual assistant by repeating misunderstood words. When a user repeats part of the phrase the system rewrites the original query to incorporate the correction. This rewrite allows the virtual assistant to understand the original query successfully. We present an end-to-end 2-step attention pointer network that can generate the the rewritten query by merging together the incorrectly understood utterance with the correction follow-up. We evaluate the model on data collected for this task and compare the proposed model to a rule-based baseline and a standard pointer network. We show that rewriting the original query is an effective way to handle repetition-based recovery and that the proposed model outperforms the rule based baseline, reducing Word Error Rate by 19% relative at 2% False Alarm Rate on annotated data.

1. Introduction

Virtual assistants that use Automatic Speech Recognition (ASR) rely on the accuracy of the transcript to understand the user's intent. However, even though ASR accuracy has improved, recognition errors are common. Errors typically occur due to both noise and the ambiguity of language. Xiong et al. [1] found the error rate of professional transcribers to be 5.9% on Switchboard, which is a dataset of conversational speech. Specifically, for out-of-vocabulary words, like names, ASR systems often make mistakes. Recognition errors lead to misunderstandings of the user's intent, which results in unwanted behaviour. Recognition errors can be mitigated by detecting that an error has occurred [2, 3]. If this is the case, the virtual assistant might initiate a recovery strategy. Typical recovery strategies include asking clarifying questions or asking the user to repeat a certain part of the request [4].

If the system fails to detect a recognition error and asks the user to correct, it may respond nonsensically or take an action the user did not intend. In response, the user might try to alleviate the misunderstanding. The most common way to do this is repeating (part of) the utterance [5]. We call this repetition-based recovery. Substantial work exists on detecting if such a repetition-based correction occurred using prosody [6, 7, 8], n-best hypothesis overlap [9], phonetic distances [10] or a combination of the above [11].

In this paper, we focus on automatically correcting the error that was made through Query Rewrite (QR). Query Rewrite is used in information retrieval systems [12, 13] and smart assistants [14, 15, 16]. We generate a corrected transcription of the incorrectly understood utterance. A Natural Language Understanding (NLU) system can then parse this rewritten query so that the system can recover from the error. QR is a generalizable way of handling repetition-based recovery, because it makes no assumptions about a specific grammar the user should be following. This is especially important for virtual assistants that handle a wide range of requests. Previously proposed systems for repetition-based recovery [17, 10] assume a specific grammar or task making them hard to use for a virtual assistant.

We propose a model that can generate a rewritten query by merging together the incorrect first utterance and the correction follow-up utterance. We can then use the rewritten query to re-estimate the user's intent and act upon it. The inputs to the model are Acoustic Neighbour Embeddings (ANE) [18] of the words in the utterances. ANEs are embeddings that are trained in such a way that similar sounding words are close to each other in the embeddings space. The ANE model takes in a word as a sequence of graphemes and produces a word embedding. With these embeddings, the model can infer which words sound similar and thus could have been confused by the ASR. Wang et al. [19] rewrite single turn utterances by taking phonetic inputs from the ASR model to handle entity recognition errors.

Our proposed model is an encoder-decoder [20] network. The decoder is a modified pointer network [21] that takes two input sequences, one for the incorrect first utterance and one for the correction follow-up utterance. A pointer network is an attention based [22] model that outputs words from a variable-sized dictionary. A pointer network is necessary because the model only considers words occurring in the two input utterances as potential outputs. We modify the standard pointer network to a 2-Step attention pointer network, which is tailored to this problem. The pointer network first attends over the first turn, then attends over the second turn and finally selects between the two attention results.

The model is trained using synthetic data generated based on a transcribed speech dataset. Using the ASR transcription and the reference transcription, a repetition-based recovery is generated. This data generation method allows the rewrite
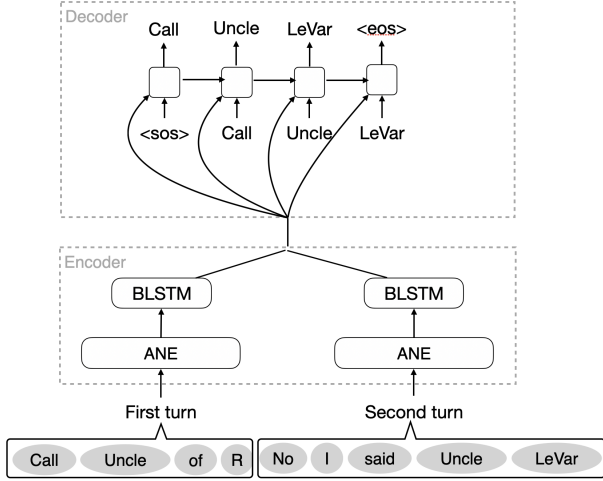
Figure 1: *A diagram of the proposed model. The words are embedded using the ANE model. These embeddings are then encoded using a BLSTM. The decoder generates the rewritten query one word at a time using a pointer network.*
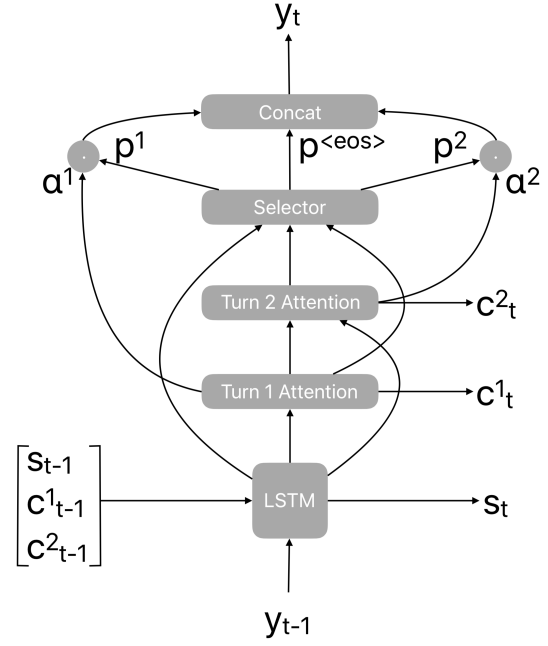

Figure 2: *A schematic of one step of the 2SA decoder. The LSTM updates the state. The utterance 1 Attention looks in the first utterance for the relevant word. The utterance 2 Attention then looks into the second utterance for a similar sounding alternative. Finally, the selector selects between the two candidates or the end-of-sequence label.*

model to be trained using existing commonly available data resources.

The contributions of this paper are threefold:

1. We propose QR as a generalizable way of handling repetition-based recovery.

2. We propose 2-Step Attention (SA2) pointer network for this task.

3. We propose a method to generate training data based on transcribed speech.

The remainder of this paper is organised as follows. First, in Section 2, we describe the proposed model, and in section 3, we explain how we generate training data to train this model. In Section 4, we describe the models we used as baselines. Next, in Section 5, we present the data we used for the experiments, the metrics we used to measure the performance and the model hyper-parameters. In Section 6, we discuss the results. Finally, in Sections 7 and 8, we formulate future work and conclusions.

2. Model

Our proposed model takes two utterances as inputs: the potentially incorrect first utterance and the correction followup utterance. The output is a rewritten query. The rewritten query is represented as a sequence of pointers to words in the incorrect first utterance and the second utterance. To clarify take the following example:

- **First utterance**: Call Uncle LeVar
- **ASR transcription**: Call Uncle of R
- **Second utterance**: No, I said Uncle LeVar
- **Model output**: 1-1 1-2 2-5

1-1 represents first word in first utterance (Call), 1-2 represents second word in first utterance (Uncle), 2-5 represents the fifth word in the second utterance, LeVar in this example. The model thus replaces "of R" with "LeVar", keeping the "call" verb. Some words can occur in both sentences, such as "Uncle". We

select the first-turn word as target during training. Figure 1 depicts a diagram of the proposed model. Each component is discussed in detail in the following subsections.

2.1. Encoder

We encode each utterance into a sequence of context-aware word vectors. We embed the words into ANEs. The ANE model is an LSTM that takes in the sequence of graphemes that make up the word and produces an embedding vector [18]:

$$e = \text{ANE}(G), \tag{1}$$

where e is the ANE and G is the grapheme sequence. Finally, we pass the ANEs through a bidirectional LSTM (BLSTM) [23] to create context-aware word representations.

2.2. Decoder

A schematic of one step of the 2SA decoder is found in Figure 2. The 2SA decoder is a modified pointer network [21] and takes the following inputs:

- y_{t-1}: The word from the previous decoder step. This vector is taken out of the word-level representation discussed in Section 2.1.

- s_{t-1}: The decoder LSTM state from the previous decoder step.

- c_{t-1}^1: The first utterance attention context from the previous decoder step.

- c_{t-1}^2: The second utterance attention context from the previous decoder step.

The output of the decoder step is the next word of the output sequence, which is chosen from the words in the two input utterances.

First, we feed all the inputs into the LSTM to obtain the updated decoder LSTM state s_t. This updated state is used to query the first utterance using an attention mechanism [22]. The attention mechanism gives a vector α^1 that contains a weight for each word in the first utterance. These attention weights are used to compute the context vector c_t^1, which is a weighted average of the context-aware word vectors using α^1 as weights. We then query the second utterance using the context from the first utterance and the LSTM state in the same way to get the attention weights α^2 and second utterance context vector c_t^2.

The intuition of the above is as follows: Both attention mechanisms look into an utterance for a candidate word to output next. We start by looking into the first utterance, because the corrected transcription is expected to be similar to the first utterance, so it is relatively easy to know which part to attend to next. Once we have a candidate word from the first utterance, we look into the second utterance for a similar sounding word that could act as a replacement.

Once the attention mechanisms have found a candidate for each utterance, the selector chooses which word to output. The selector receives the LSTM state and both attention contexts as input and outputs three probabilities:

1. p^1: Probability that the candidate from the first utterance is correct
2. p^2: Probability that the candidate from the second utterance is correct
3. $p^{\langle eos \rangle}$: Probability for ending the sequence

We then create a probability distribution over all words in both utterances and the $\langle eos \rangle$-label by multiplying each utterance probability with the respective attention weights:

$$p = \begin{bmatrix} p^1 \alpha^1 \\ p^2 \alpha^2 \\ p^{\langle eos \rangle} \end{bmatrix}, \qquad (2)$$

where p is the probability distribution. We then select the next word y from this distribution.

2.3. Inference

During training we use the ground truth output sequences as inputs to the decoder. For inference we do beam search to find the most likely output sequence, where we keep generating words until an end-of-sequence token is generated. We also enforce a strict left-to-right attention policy in both attention mechanisms. This means that the attention mechanism can only attend to words to later in the utterance than the last word selected from this utterance.

2.4. Classifier

For a virtual assistant, the vast majority of interactions are not repetition-based recoveries. For these cases, the model should not rewrite the original query. To determine whether a rewrite is necessary, we add a classifier that classifies a rewrite as either necessary or not necessary. In this paper, we compute a phonetic weighted Levenshtein distance between the original request and the rewrite. The costs of the edits in the Levensthein distance are based on a phonetic edit distance that was computed based on the confusability of phones in the ASR. If this distance is lower than some threshold, the rewrite is deemed necessary.

3. Training data generation

To train the model, we synthetically generated repetition-based recovery interactions based on a transcribed speech dataset. We first run all the utterances in the dataset through the ASR to get ASR transcriptions. For the cases, where the ASR transcription contains an error we find the substitution the ASR made and generate a repetition-based recovery. In the example in Section 2, the ASR made the substitution "LeVar" → "of R". Using this substitution, a repetition-based recovery is generated using the following heuristics:

- A random number of words to the left and right of the error are included in the recovery. For 85% of cases, this is 0 words, for 10% this is one word and for 5% this is 2 words.

- For 10% of repetition-based recoveries, a prefix is included. This prefix was randomly selected for a list of common prefixes, like "No, I said".

With these heuristics some of the repetition-based recoveries that can be generated for the above example are:

- LeVar
- Uncle LeVar
- No, I said uncle LeVaR

The model is trained using the ASR transcription and generated repetition-based recovery as input and the reference transcription as the target rewrite.

4. Baseline

We use two baselines to compare the results of our proposed model: one rule-based model based on phonetic alignments and one more standard pointer network similar to our proposed model.

The rule-based model generates a list of all possible n-gram pairs of both utterances for potential replacements. We compute a phonetic edit distance for all pairs and generate the corrected transcription by replacing the n-gram in the first utterance with the n-gram from the correction follow-up. We compute the phonetic edit distance using a phone confusion matrix and normalise that distance by dividing by the number of phones in the first utterance n-gram.

For the example given in Section 2, the replacement with the smallest phonetic edit distance would be for replacing "Uncle of R" with "Uncle LeVar".

Furthermore, we implement an unmodified pointer network [21] to compare with our proposed 2SA pointer network. We use the same ANE vectors as input instead of word embeddings.

5. Experiments

5.1. Dataset

Both neural network models are trained on an internally collected dataset described in Section 3. Using the transcribed first turn, we generate the second turn and the target rewrite. This leads to about 350k turn pairs. For evaluation, we employ

a fully annotated test set with 9.2k turn pairs. The annotators transcribe both turns and indicate whether the user intended to correct in the second turn.

Table 1: *In this table we show the evaluation dataset statistics. The human-graded evaluation dataset reflects traffic, where majority of follow-ups are not correction.*

	count	proportion
Rewrite correction	450	4.8%
No correction	8840	95.2%

5.2. Metrics

We compute the word error rate reduction (WERR) by comparing the WER of the original first turn utterances with the WER of the rewrites.

$$WERR = 1 - \frac{WER\_rewrite}{WER\_original} \quad (3)$$

We measure WERR only on turn pairs that have been annotated as correction, as a rewrite is not required for the others. We additionally compute false alarm rates (FAR) at a range of classifier thresholds. FAR is identical to false positive rate and it's the proportion of unnecessary rewrites.

5.3. Training setup

The baseline pointer network has 1 encoder and decoder BLSTM layer with 128 hidden size. We train it using learning rate 0.0003 with Adam and batch_size 32. For our proposed model we employ 2 encoders (one for each turn) and 1 decoder with the same dimension, but modified attention head. We train it with batch size 128 and learning rate 0.0001 with Adam. We use pretrained ANEs that are fixed during training.

While the 2SA has twice as many encoder parameters, each encoder only sees half of the data as they only receive their respective turn. For the pointer network, we concatenate both turns to generate a rewrite.

6. Results

We measure the WERR of our three proposed approaches on the test set at all classifier thresholds in Figure 3. We zoom in on the WERR results in Table 2.

The rule based baseline achieves a higher maximum WERR, but at a higher FAR. Both machine learned models give high WERR early on before tapering off. We compare the pointer-network and the 2SA network and note the pointer-network has lower FAR for the same WERR, but lower maximum WERR overall, showing a FAR/WERR tradeoff between the two model approaches.

We note the rule-based rewriter uses the same edit distance minimizing rules on positive and negative pairs. Our classifier uses edit distance to score the rewrite. Given the unbalance of our evaluation dataset and real life traffic, it requires a low classifier threshold (implying high FAR) to be feasible. The machine learned do a rewrite when it makes sense to do so, and output random rewrites otherwise. These rewrites will have a high edit distance and be filtered by the classifier.
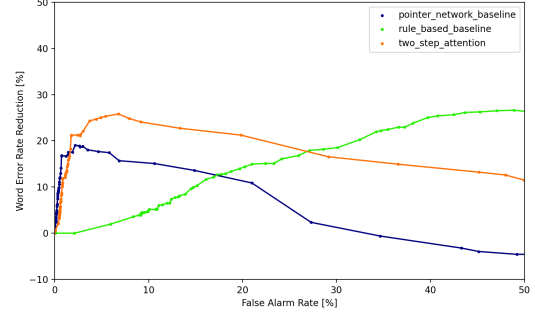

Figure 3: *On the x-axis we show the False Alarm Rate. On the y-axis we show the Word Error Rate Reduction. (Higher is better) Each point represents a classifier threshold. Low threshold corresponds to high FAR and vice versa. We compare the rule-based baseline with two neural network approaches.*

Table 2: *We evaluate the model performance on the test set. We find the best possible WERR and show its corresponding FAR*

Model	Max WERR	FAR
rule-based baseline	26.60%	48.88%
pointer-network	19.03%	**2.15**%
2 step attention pointer-network	25.80%	6.77%

7. Future Work

The user can try to help correct errors by providing extra information to the system. For example, the user might describe an entity that was misrecognized. This is not captured by pure query rewrite. In future work, one could investigate modifying the task to incorporate this extra information.

The proposed system cannot correct the error if the ASR repeats the error in the correction followup. However, it is possible that the correct hypothesis is present in the n-best list of the speech recognizer. By combining information from both utterances, it might be possible to surface the correct hypothesis.

8. Conclusion

We propose a system that allows the user to correct recognition errors through repetition. The model handles this repetition-based recovery by rewriting the original query to incorporate the follow-up correction. The rewritten query is then scored with an edit distance based classifier for thresholding. We propose a 2-Step Attention pointer network and show that it outperforms both a standard pointer network and a rule-based baseline. We propose a way to generate synthetic training data using only transcribed data and an ASR system to train our machine learned models.

9. References

[1] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016.

[2] A. Ogawa and T. Hori, "Error detection and accuracy estimation

in automatic speech recognition using deep bidirectional recurrent neural networks," *Speech Communication*, vol. 89, pp. 70–83, 2017.

[3] J. Fayolle, F. Moreau, C. Raymond, G. Gravier, and P. Gros, "CRF-based combination of contextual features to improve a posteriori word-level confidence measures," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010, pp. 1942–1945.

[4] G. Skantze, "Galatea: A discourse modeller supporting concept-level error handling in spoken dialogue systems," in *Recent Trends in Discourse and Dialogue*. Springer, 2008, pp. 155–189.

[5] M. Swerts, D. Litman, and J. Hirschberg, "Corrections in spoken dialogue systems," in *Sixth International Conference on Spoken Language Processing*, 2000, pp. 615–618.

[6] D. Litman, J. Hirschberg, and M. Swerts, "Characterizing and predicting corrections in spoken dialogue systems," *Computational linguistics*, vol. 32, no. 3, pp. 417–438, 2006.

[7] J. Hirschberg, D. Litman, and M. Swerts, "Prosodic and other cues to speech recognition failures," *Speech Communication*, vol. 43, no. 1-2, pp. 155–175, 2004.

[8] L. J. Stifelman, "User repairs of speech recognition errors: An intonational analysis," *Manuscript*, 1993.

[9] N. Kitaoka, N. Kakutani, and S. Nakagawa, "Detection and recognition of correction utterance in spontaneously spoken dialog," in *Eighth European Conference on Speech Communication and Technology*, 2003, pp. 625–628.

[10] J. Lopes, G. Salvi, G. Skantze, A. Abad, J. Gustafson, F. Batista, R. Meena, and I. Trancoso, "Detecting repetitions in spoken dialogue systems using phonetic distances," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015, pp. 1805–1809.

[11] G. Skantze and J. Edlund, "Early error detection on word level," in *COST278 and ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction*, 2004.

[12] R. Musa, X. Wang, A. Fokoue, N. Mattei, M. Chang, P. Kapanipathi, B. Makni, K. Talamadupula, and M. Witbrock, "Answering science exam questions using query reformulation with background knowledge," in *Automated Knowledge Base Construction (AKBC)*, 2019. [Online]. Available: https://openreview.net/forum?id=HJxYZ-5paX

[13] D. Demszky, K. Guu, and P. Liang, "Transforming question answering datasets into natural language inference datasets," *CoRR*, vol. abs/1809.02922, 2018. [Online]. Available: http://arxiv.org/abs/1809.02922

[14] Z. Chen, X. Fan, and Y. Ling, "Pre-training for query rewriting in a spoken language understanding system," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7969–7973.

[15] A. Roshan-Ghias, C. S. Mathialagan, P. Ponnusamy, L. Mathias, and C. Guo, "Personalized query rewriting in conversational ai agents," *ArXiv*, vol. abs/2011.04748, 2020.

[16] P. Rastogi, A. Gupta, T. Chen, and L. Mathias, "Scaling multi-domain dialogue state tracking via query reformulation," in *NAACL-HLT*, 2019.

[17] H. Sagawa, T. Mitamura, and E. Nyberg, "Correction grammars for error handling in a speech dialog system," in *Proceedings of HLT-NAACL 2004: Short Papers*, 2004, pp. 61–64.

[18] W. Jeon, "Acoustic neighbor embeddings," 2020.

[19] H. Wang, S. Dong, Y. Liu, J. Logan, A. K. Agrawal, and Y. Liu, "ASR error correction with augmented transformer for entity retrieval," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, H. Meng, B. Xu, and T. F. Zheng, Eds. ISCA, 2020, pp. 1550–1554. [Online]. Available: https://doi.org/10.21437/Interspeech.2020-1753

[20] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[21] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2692–2700.

[22] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.