Search based Self-Learning Query Rewrite System in Conversational AI

Xing Fan
Alexa AI, Amazon
USA
fanxing@amazon.com

Eunah Cho
Alexa AI, Amazon
USA
eunahch@amazon.com

Xiaojiang Huang
Alexa AI, Amazon
USA
xjhuang@amazon.com

Chenlei Guo
Alexa AI, Amazon
USA
guochenl@amazon.com

ABSTRACT

Query rewriting (QR) is an increasingly important technique for reducing user friction in a conversational AI system. User friction is caused by various reasons, including errors in automatic speech recognition (ASR), natural language understanding (NLU), entity resolution (ER) component, or users' slip of the tongue. In this work, we propose a search-based self-learning QR framework: User Feedback Search based Query Rewrite system (UFS-QR), which focuses on automatic reduction of user friction for large scale conversational AI agents. The proposed search engine, operating on both global user and individual user level, leverages semantic embedding, NLU output, query popularity and estimated friction statistics into the retrieval and ranking process. In order to construct the index and train the retrieval/ranking models, we adopt a self-learning based method by utilizing implicit feedback, learned from users' historical interactions. We demonstrate the effectiveness of the UFS-QR system, trained without any annotated data, through offline and online A/B experiment on Amazon Alexa user traffic. To the best of our knowledge, this is the first deployed self-learning and search-based QR system for the general task of automatic friction reduction in conversational AI.

KEYWORDS

Query rewrite, Spoken dialogue system, Search, Self-learning

ACM Reference Format:
Xing Fan, Eunah Cho, Xiaojiang Huang, and Chenlei Guo. 2021. Search based Self-Learning Query Rewrite System in Conversational AI. In *2nd International Workshop on Data-Efficient Machine Learning (DeMaL), Virtual.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With the increased popularity of virtual assistant agents such as Alexa, Cortana and Siri, millions of users interact with spoken dialogue systems on daily basis. Unavoidable, some interactions result in *friction*. There are mainly two types of errors that lead to *friction*. The first type is system error, which refers to the errors accumulated throughout the model pipeline, including Automatic Speech Recognition (ASR), Natural Language Understanding (NLU), dialogue Management (DM), etc. For example, an ASR error can lead to a wrong recognition "play maj dragons" instead of user's intended "play imagine dragons". The second type is user ambiguity, such as user's slip of the tongue or using abridged language while making the query. Once there is a friction, it often requires longer engagement with the user to clarify the query and may lead to possible abandonment of the task. Query rewrite (QR) aims to automatically map a user query to another form, so that the dialogue system can be more robust.

Previous efforts for QR include a spelling correction model for ASR [8], followed by an embedding based method relying on human annotated data [11]. Leveraging user engagement signals into the NLU model training [15] was explored as well. These methods focused on reducing the errors in a specific component such as ASR or NLU. Thus, they often lack the flexibility or generalization ability to capture various errors originated from dialogue system pipeline or the user.

In this paper, we present User Feedback Search based Query Rewrite (UFS-QR) system. We introduce a search component inside the conversational AI agent to automatically resolve different types of system errors by leveraging users' past interaction history. Documents are constructed using past interactions between the users and the agent. Specifically, we introduce a) a global layer where the documents are constructed by aggregating all users' historical interactions and b) a personalization layer where the documents are constructed using individual user's historical interactions. This is to reflect the nature of friction; while some frictions occur globally for many users (e.g. "tooth or dare" as a mis-cognized query of "truth or dare"), it is also often expected to provide a personalized service to correctly fulfill each individual user's request. For example, for a query "play imagine", certain users may refer to the song by the artist John Lennon, while others meant the artist Ariana Grande. Figure 1 shows the overview of UFS-QR system. Each global and personalized layer utilizes separated indexes, followed by separated retrieval and ranking components. Rewrites from each layer are arbitrated and the final top 1 rewrite is used in the downstream system.

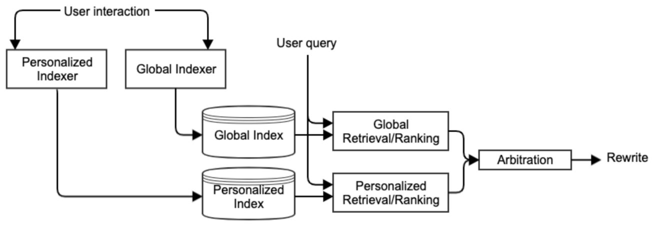

Figure 1: System overview of the UFS-QR system

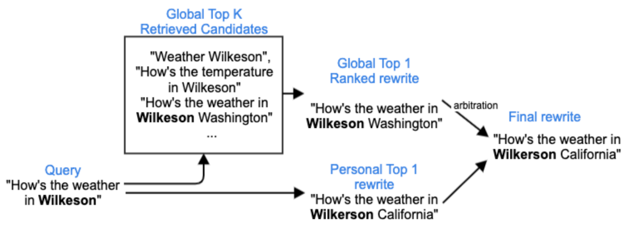
Global Top K Retrieved Candidates
"Weather Wilkeson", "How's the temperature in Wilkeson" "How's the weather in Wilkeson Washington" ...
Global Top 1 Ranked rewrite
"How's the weather in Wilkeson Washington"
arbitration
Final rewrite
"How's the weather in Wilkerson California"
Query
"How's the weather in Wilkeson"
Personal Top 1 rewrite
"How's the weather in Wilkerson California"

Figure 2: Example of the global/personalized UFS-QR.

UFS-QR is a self-learning system. Training data for our system does not require human annotation, and are obtained using users' implicit feedback. The implicit feedback data is collected through predefined heuristics to utilize user's own reformulation of queries. For example, when ASR recognizes a user's query with an error, as "play ambient mean", this will lead to friction, with a response from the agent, "Sorry, I couldn't find ambient mean". Some users may abandon the task, while a few users may continue trying by enunciating. When the utterance is correctly recognized ("play envy me"), this will lead to the action from the agent, "Here is Envy Me, by Calboy". We collect such data to train the models inside the search system. Beside implicit feedback data, we also utilize agent response quality metrics from the spoken dialogue system. The metrics are generated from rules (e.g. whether the agent response is "Sorry, I don't know") as well as a machine learning model, which provides an estimated defect prediction for each interaction [14]. The metrics are incorporated as input features to the modeling components to impact the search decisions. By utilizing users' historical interactions and associated metrics, UFS-QR is able to correct both system and user errors in a proactive way without requiring annotation data. Figure 2 shows an example query and its rewrite through UFS-QR.

2 RELATED WORK

Query rewriting in web search refers to the process of reformulating an original input query to a new query, aiming to achieve better search results. It is a critical component in modern search engines [6, 9, 19, 24]. Motivated by its success, many voice search systems utilize this component to correct ASR related errors [8, 23, 25]. However, there are unique challenges for adopting QR in a large scale spoken dialogue system. For example, such system often relies on voice-user interface to interact with users. As confirmation with users by introducing an extra turn is typically considered as a friction, one needs to maintain high precision for the QR system.

In order to automatically estimate the turn-based defect rate in a voice-based virtual assistant system, different modeling methods were proposed in the past such as [1, 14, 17, 21]. Such machine generated metrics play an important role to enable self-learning and automatically reduce friction in a spoken dialogue system.

Self-learning based friction reduction for a large scale spoken dialogue system was studied in Ponnusamy et al. [18]. The authors proposed an absorbing Markov Chain model as a collaborative filtering mechanism to mine user's own reformulation patterns. Once the patterns are discovered, rewrite pairs are extracted. At runtime, if a query is an exact text match with a friction query, a rewrite will be triggered using the offline mined pairs. Different from the Markov chain model, Chen et al. [4] proposed a retrieval model for QR task, utilizing a query encoder that incorporates contextual language modeling pre-training. Bonadiman et al. [2] is another study that focuses on friction reduction using a question paraphrase retrieval. Similar to Chen et al. [4], authors adopted a semantic encoder model and proposed a smoothed deep metric loss to handle the noisy labels. Different from Ponnusamy et al. [18] and Chen et al. [4], Bonadiman et al. [2] focused on question answering task.

The difference of UFS-QR system to the Markov chain model is that our system allows improved flexibility and generalization over input queries. While the work in Ponnusamy et al. [18] also leveraged self-learning, it has a limitation that an input query should be an exact text match of a previously observed friction utterance. Our approach offers further expansion on this by utilizing the search based methods and metrics such as similarity measures. In addition, compared to Chen et al. [4], we further explore the importance of a ranking layer leveraging a neural feature extractor and a tree model.

3 SYSTEM OVERVIEW

Conventional spoken dialogue systems largely contain five components: ASR, NLU, DM, a natural language generation (NLG) system, and a text-to-speech (TTS) system. When a user interacts with his/her device, the audio is passed through the ASR and transcribed into a text. We refer this ASR hypothesis as 'query'. The query is passed through the NLU component followed by the DM component. It extracts the domain, intent, entity information and decides the action to execute. After that, agent response is decided through the NLG component. Finally, the TTS generates the audio speech response, which is sent back to the device to finish one interaction loop. All metadata associated with each of the above systems are anonymized and logged asynchronously to an external database.

The proposed UFS-QR system first takes the query as an input to the search stack. We use the confidence score from the inference model to decide if the returned rewrite should be triggered or not. The triggering threshold is decided empirically from a hold out set. The rewrite is passed to the NLU and the original data flow is restored. We demonstrate the design in Figure 3. Similar to Ponnusamy et al. [18], a blocking mechanism to disable any bad rewrites is deployed together with UFS-QR. For this, we leverage Z-test to compare the friction rate of original queries and their rewrites. We adopt a machine learning model proposed in [14] to automatically estimate the friction rate.

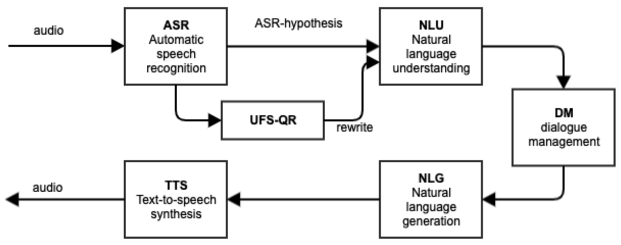

Figure 3: UFS-QR in spoken dialogue system.

In general, a search system comprises of two stacks operating sequentially: retrieval layer and ranking layer. From a great amount of documents, the retrieval layer aims to retrieve a set of relevant documents with low latency and computational cost. From the retrieved documents, ranking layer targets to rank the most desired documents to the top, often leveraging more complex algorithms and models. Similarly, the UFS-QR system contains: a) index, constructed from an offline process, b) retrieval layer, that performs approximate nearest neighbor search, c) ranking layer, which ranks the rewrite candidates and returns the top 1 rewrite as the final rewrite.

4 UFS-QR INDEX

4.1 Global Index

Global index provides rewrite candidates extracted from all user interactions for the modeling components (retrieval and ranking). Our global index is generated from aggregated, anonymized historical interactions between the users and the virtual assistant agent within a period of time (e.g. 30 days).

NLU hypothesis based document The document in the global index is constructed using NLU hypothesis information. Using NLU hypothesis information, we aim to represent a group of queries sharing a similar goal in a dialogue system. NLU hypothesis is generated from the NLU component in the spoken dialogue system, and can be represented in the following format "domain | intent | slot_type:slot_value". For example, given a query "play bad blood by taylor swift", we have an NLU hypothesis of "Music | PlayMusic | SongName:bad blood | ArtistName:taylor swift". In each document, we list all queries that lead to the same NLU hypothesis, gathered from anonymized user interaction data. As we do not rely on any human annotation process for the index generation, it is possible that the NLU hypothesis may contain errors. Compared to the documents constructed with query information only [2], this design offers an advantage to leverage both head and tail queries that represent the same goal and to be more robust to noise.

Friction information Our index incorporates historical friction rate of each query, obtained from IQ-Net, a DNN model that predicts interaction-level dialogue quality [14] and rule-based defect prediction, e.g. agent response starts with "sorry". Each query has information on the total impression (referred as *impression*, total occurrences of the query within a period of time) and the defect impression (referred as *defect*, total occurrences of the query when the given metric model classifies it as friction). The occurrence is accumulated from all users.

Combine existing rewrites We further expand the design to include two types of documents. The first type is *ASR*, where we take the NLU hypothesis of the given ASR 1-best. The second type is *AUS* (Alternative Utterance Service). AUS rewrites are obtained from hand-crafted rules and following previous approaches [18]. While AUS rewrites operate in a simple text match form, our purpose is to further generalize on different queries by incorporating them into the index. For example, given a query and existing AUS rewrite pair of "play the alphabet song" - "play the abc song", our goal is to transform other variances of the query (e.g. "do alphabet song") into the rewrite. Despite the query is not an exact match of previously observed friction query, UFS-QR retrieval layer is able to retrieve the most relevant rewrite candidates from the index. We list the original queries under each document and use a suffix (AUS or ASR) for the NLU hypothesis. Figure 4 provides examples.

We generate the global index from one month of all historical interactions and apply impression based pruning (a cut-off threshold) to remove noise.

4.2 Personalized Index

Personalized index follows the similar format of the global UFS-QR index shown in Figure 4, but is built for each user, leveraging individual interaction history. We leverage the *impression* and *defect* information, from both global and personalized level. For example, a query "turn on the moon light" may result in a successful user experience for the user who has a lamp named "the moon". However, for other users, this utterance might be user's slip of the tongue for "moonlight sonata", leading to unsuccessful user experience.

Selecting utterances by user-level *impression* and *defect*, personalized index consists of successful queries spoken by the user[1] within a recent time period (i.e. 30 days). In general, we observe more correction opportunities when utilizing a longer time period for index building (e.g. 90 days). In order to balance the runtime latency constraints, index storage size as well as the opportunity size, we choose 30 days for the time period for personalized index. Compared to the global index, the personalized index covers more utterances from the tail distribution. In order to reflect recency and users' potential preference change, we update this index in a regular cadence (i.e. daily). To better define the search space and consider runtime aspects, we limit the personalized index size, by selecting top l history utterances based on *impression* and recency.

5 RETRIEVAL LAYER

We frame the problem in retrieval layer as described in Henderson et al. [10]. Let x be the input query and y the candidate rewrite. As shown in Eq. 1, joint probability $P(x, y)$ is represented by a neural network scoring function $S(x, y)$.

$$P(y|x) = \frac{P(x,y)}{\sum_k P(x,y_k)}, S(x,y) = \cos(\mathbf{h}, \mathbf{r}) \qquad (1)$$

In order to support a large number of indexed rewrite candidates, we focus on a system based on query embedding [4] and select the most relevant candidates. A neural encoder learns to capture latent

[1]All user information was in an anonymized format.

```
{
  "Music|AddToPlayQueueIntent|ASR": {
    "queries" : {
      "play the queue up" : {"Impression": 79, "Defect": 10},
      "play queue up" : {"Impression: 27, "Defect": 5},
      "play queue" : {"Impression": 119, "Defect": 10},
    }
  },
  "Music|AddToPlayQueueIntent|ArtistName:maroon five|SongName:girls|ASR": {
    "queries" : {
      "play girls by queue by maroon five" : {"Impression": 143, "Defect": 20},
  },
  "Books|ReadBookIntent|BookName:firefight|AUS": {
    "queries" : {
      "read firefly" : {"Impression": 21, "Defect": 10},
      "can you read firefly" : {"Impression": 5, "Defect": 0},
    },
    "aus_replaced_request": {
        "read firefight"
    },
}
....
```

Figure 4: Simulated examples of documents in the index.

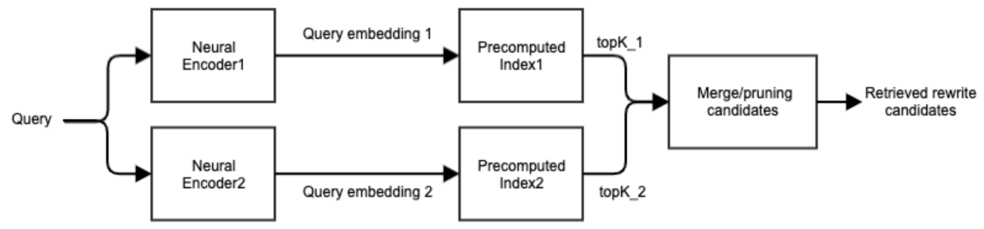

Figure 5: Multi-model architecture in retrieval layer.

syntactic and semantic information for the query and the rewrite candidates. The final similarity comparison is conducted afterwards. Scoring function is shown in Eq. 1, where *cos* represents cosine distance. $\mathbf{h}$ is the embedding of x and $\mathbf{r}$ is the embedding of y, generated by passing the input x or y through the neural encoder respectively. The neural encoder is trained so that the query's embedding is close to its corresponding rewrite in the projected space. While the encoder can be any type of sequence-to-vector transformations, we considered a DNN-based encoder [12] and a CNN-based one [22] for run-time efficiency. During training, given pairs of (x, y), we calculate $P(y|x)$ as follows:

$$P(y|x) \approx \frac{\exp\{S(x,y)\}}{\sum_{\tilde{k}\in K} \exp\{S(x,y_k)\} + \exp\left\{S(x,y_{k\_BM25})\right\}}, \quad (2)$$

Ideally, K should be all rewrite candidates. In this paper, we construct them from each mini-batch for training efficiency and use cross entropy as the loss function. Another key component is the k-Nearest Neighbour (kNN) index for a fast top K retrieval[2]. Prior to the inference time, all rewrite candidates are encoded and added to the FAISS index. During the inference time, each query is encoded and FAISS returns the top K relevant rewrites.

We adopt a multi-model architecture (Figure 5), to promote the diversity of the retrieved candidates. A single query is passed to two different encoders. Each DNN and CNN encoder retrieves top K rewrites separately. The candidates are first sorted separately and joined in an interleaving way. Starting from the top of the joined list, we examine the NLU hypothesis of the candidates; candidates whose NLU hypothesis has been seen before are removed. The process stops if we obtain the predefined K candidates or we examined through the interleaved list. This merge/prune process helps to remove any redundant candidates from the retrieval layer, improving the coverage of candidates. Final rewrite candidates are passed to the ranking layer with their corresponding NLU hypothesis.

In our preliminary experiments, we experimented with utilizing up to 4 retrieval models with different configurations (e.g. different input sequences, architecture, etc.) in the retrieval layer, and observed gain in the retrieval performance. Considering the runtime constraints such as memory footprint and latency, we choose the aforementioned DNN and CNN models as our retrieval models. The DNN model in the retrieval layer takes character-level trigram as

[2]We use FAISS for efficient retrieval [13].

input. We use three layers of fully connected MLP with 512 hidden layer size. The final embedding size for both DNN and CNN is 300.

6 RANKING LAYER

The ranking component takes query x, top K candidates Y from the retrieval layer, and their corresponding NLU hypothesis Y_{NLU} as input. It has two main model components, a neural feature extractor and a Gradient Boosted Decision Tree (GBDT). We choose the hybrid model architecture for the following considerations: 1) We want the flexibility of the ranking layer to further incorporate richer semantic information (e.g. leveraging pre-trained contextual language embeddings [5, 7]); 2) Model should be easily extended to incorporate various discrete or continuous features for both global and personal level; 3) Neural ranking models are typically sensitive to the training data distributions. We believe fusing the neural features with other features for the tree model helps the generalization.

6.1 Neural Feature Extractor

The neural feature extractor is a deep learning model that provides hidden embedding as features for GBDT. Given the query text x, we first pass the sequence of tokens through an encoder and obtain a sequence of hidden representation $\mathbf{h}$. For the rewrite hypothesis y_{NLU}, we serialize it into a text sequence. It is passed through the encoder the same way as the query text. We obtain $\mathbf{r}$, the rewrite hidden representation.

In this paper, we use MLP for the encoder due to the limited latency budget. We obtain the hidden representations $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, ..\mathbf{h}_N$ for query and $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, ...\mathbf{r}_M$ for the rewrite hypothesis[3]. We then generate a Bag Of Embedding (BOE) for each side as follows:

$$\bar{\mathbf{u}} = \sum_i \mathbf{u}_i \tag{3}$$

Note that $\mathbf{u}$ can be either query side ($\mathbf{h}$) or rewrite hypothesis side ($\mathbf{r}$). With the BOE of query ($\bar{\mathbf{h}}$) and rewrite NLU hypothesis ($\bar{\mathbf{r}}$), we conduct single-head cross attention (between query side and rewrite side), as shown in Figure 6. We generate attention weight of $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, ..\mathbf{h}_N$ by comparing them with the $\bar{\mathbf{r}}$. The context vector is generated by weighted sum of the $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, ..\mathbf{h}_N$. In this paper, we use a dot product to generate the attention weights. A context vector is generated for both query and rewrite side as $\mathbf{c}_h$ and $\mathbf{c}_r$. The final embedding extracted from either query or rewrite side is the concatenation of their BOE and context vector. We use the following transformation and feed this to the final MLP layer that outputs a ranking score.

$$score = \text{MLP}(\mathbf{f}_{enc}), \mathbf{f}_{enc} = [\widetilde{\mathbf{h}}; \widetilde{\mathbf{r}}; |\widetilde{\mathbf{h}} - \widetilde{\mathbf{r}}|; \widetilde{\mathbf{h}} \odot \widetilde{\mathbf{r}}] \tag{4}$$

$\widetilde{\mathbf{h}} = [\bar{\mathbf{h}}; \mathbf{c}_h]$ and $\widetilde{\mathbf{r}} = [\bar{\mathbf{r}}; \mathbf{c}_r]$. Different from the retrieval model training, we use margin ranking loss to train the model. Each query has a positive rewrite and a negative rewrite. The model is trained to produce a higher ranking score for the positive rewrite, compared to the negative rewrite. After the model is trained, we take the $\mathbf{f}_{enc}$ in Eq. 4 as the features extracted from the neural model and feed them to the GBDT models.

[3] N, M: token sequence length of query and rewrite NLU hypothesis

6.2 Ranking GBDT model

Besides the features extracted from the neural model, we also extract a group of conventional IR features. All features are concatenated to form an input to the GBDT model. The features can be categorized into mainly three types. **Text features** capture the text level difference between the query and rewrite, e.g. edit distance, BLEU score [16]. **Document features** provide information at document level related to the historical interaction of the given NLU hypothesis, e.g. number of queries within a document, historical friction rate of the document (both rule-based and machine-learning based). **Query-document features** carry information of the relevance of the query for the given document. Table 1 shows some of the features we used in this work[4].

For example, given the example document "Music | AddToPlayQueueIntent | ASR" in Figure 4, we extract "Number of Queries" to 3. As "Weighted Number of Queries", for "impression", we have $79 + 27 + 119 = 225$. For "defect", we have $10 + 5 + 10 = 25$. By differentiating "impression" and "defect", we allow the model to rank the hypothesis that helps to reduce user friction to the top. In total we derive a set of IR feature with dimension of 250.

Among many other loss function options, e.g. LambdaMart or LambdaRank [3], we use logistic regression to offer a calibrated score between [0, 1]. Our offline experiment showed that logistic regression model performance is comparable to the LambdaMart model's performance, especially when there is a large amount of training data available. We use the calibrated score to indicate the model's confidence and not to trigger the UFS-QR's rewrite when the model is not confident.

For the neural model in ranking layer, we first use a two-layer DNN with [500, 200] hidden size before applying the attention. We use a mini-batch size of 512 and set a maximum utterance length of 15 words. We use the Adam optimizer and set the training epoch as 20 with early stopping. For all neural models, we use an initial learning rate of 0.001. All experiments are performed using AllenNLP[5] on Nvidia V100 GPUs. We use Xgboost[6] to train the GBDT model, where we set the max tree depth to be 10, learning rate to be 0.2 and the number of trees to be 600.

7 REWRITE SELECTION AND ARBITRATION

For global UFS-QR, we take the top 1 returned rewrite from the ranking layer and use this for downstream components (e.g. NLU, ER) in the dialogue system.

In personalized UFS-QR, given user's input query, we aim to find the most similar utterance from the personalized index. As described in Section 4, personalized index contains successful user queries from each user's history. Currently, we leverage the retrieval models described in Section 5, and obtain the embeddings for input query and each utterance in user's index. The scoring function follows Eq. 1, where we check the similarity between the embedding for user's input (e.g. "turn on the moon light") and each utterance in their index (e.g. "repeat this song", "play the moonlight sonata", etc.). Current mechanism then selects the top 1 rewrite by

[4] We use log to smooth the raw count for "impression" and "defect" when calculating the weighted features

[5] https://github.com/allenai/allennlp

[6] https://xgboost.readthedocs.io/en/latest/

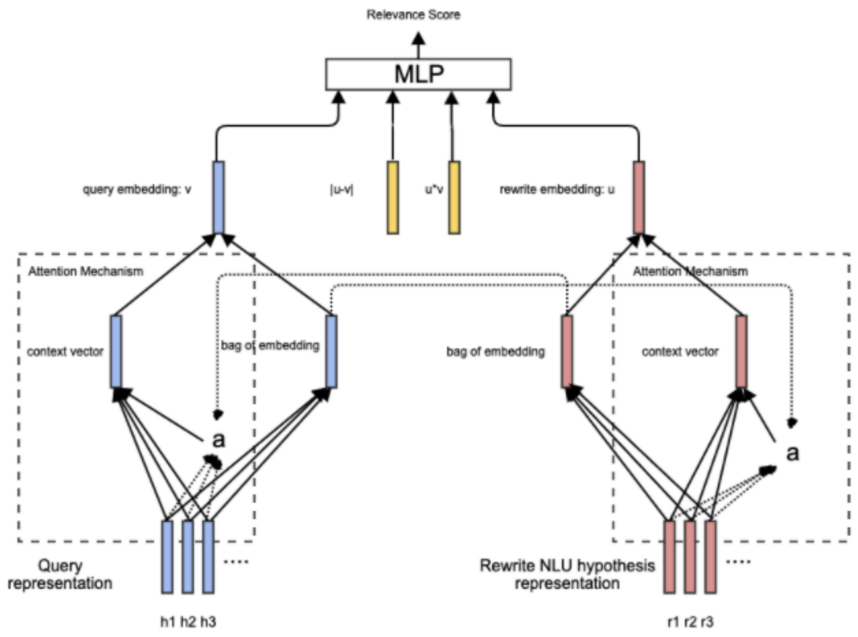

Figure 6: Model architecture for neural feature extractor of ranking layer

Table 1: Examples of IR features used. For all features that require "weight", i.e. "weighted number of tokens", "*n*-gram probability", we generate two sets of weighted features by utilizing the impression and defect respectively.

number of queries	number of queries in a document
weighted number of queries	weighted sum number of queries in a document
number of tokens	number of words in a document
weighted number of tokens	weighted number of words in a document
TF-IDF	The TF multiplied by IDF for the given word.
BM25	Okapi BM25 [20]
n-gram probability	The query probability for an *n*-gram model constructed using weighted queries in a document

this score and provides the rewrite when the score exceeds a predefined threshold θ. Similar to global UFS-QR, we consider adding a ranking component for personalization, utilizing personalized level *impression*, *defect*, and further contexts from the user.

When both global and personalized components return a rewrite candidate, we prioritize the results from personalization layer over global layer in order to support any possible personalization of QR.

8 EXPERIMENTAL SETUP

8.1 Training Data

Retrieval models Training data is obtained utilizing largely two heuristics. In the first rule, we examine any two consecutive queries from each user and find the queries that fulfill the following criteria: a) The two consecutive queries were spoken within t seconds in time window. b) The two consecutive queries have minimum edit distance shorter than d. c) Utilizing the friction detection model [14], the first query led to friction, while the second was successful. Using this rule, we aim to harvest the query-rephrase pairs from users where they repeated or rephrased the friction query. In the second rule, we utilize ASR n-best: a) first query's ASR n-best ($n > 1$) is the second query's ASR 1-best, b) same as the first rule, we maintain the rule on time window t and the friction condition on the two consecutive queries. This rule was motivated from our observation that even when there is an ASR error on the ASR 1-best, the correct recognition can often be found in their ASR n-best list. By considering the following query's ASR 1-best, we find the rephrase pairs where the first query was likely an ASR error of the second query. After careful tweaking, we chose $t = 45$, $d = 5$ to reduce noise and maintain the opportunity. The resulting 40 million pairs are used to train models for retrieval layer. The followings are examples from the training data:

- *play ambient mean* $\longrightarrow$ *play envy me* (rewrite corrects an ASR error)
- *play blues radio news* $\longrightarrow$ *play blue news radio* (rewrite rearranges words for better clarity)

Ranking models We utilize the above-mentioned 40 million pairs of training data, and obtain the positive and negative rewrites from the retrieval models. If NLU hypothesis of the reference rewrite matches the NLU hypothesis of a proposed rewrite candidate, we consider this as a positive example, and otherwise as a negative example. For example, given a query "voice room light off", top K retrievals from the retrieval models contain a correct rewrite "boys room light off", as well as an incorrect rewrite "gold room light off". The correct rewrite is assigned as a positive example of the query, while the incorrect rewrite is assigned as a negative example. We then randomly sample about 20 million total triplet "<query, positive_sample, negative_sample>" in order to train the ranking models.

8.2 Offline Evaluation

Creation of test data follows the similar procedure for training set. Test cases are randomly sampled from a time period immediately following the period from which the training data was drawn. We select the *true* rephrase pairs among them, identified by the human annotators. In order to reflect runtime impact, we select test cases that the baseline model (i.e. Markov chain based model [18]) cannot provide a rewrite for. As the Markov chain based model utilizes collaborative filtering, it often cannot capture rewrite patterns of low impression. The resulted evaluation set contains 16k test cases. We compare the global UFS-QR system performance with the Deep Structured Semantic Model (DSSM) in their default configuration [12]. For the personalized UFS-QR system, we select potential rephrase pairs that fulfill the following conditions: a) for the two consecutive utterances, the second turn utterance y is followed by the first turn utterance x within a short time window (e.g. 30 seconds) and their minimum edit distance is below an empirically chosen threshold, b) we can observe y in the user's history with successful interaction indicated by the friction estimated model. These samples are further annotated by human annotators to only select the *true* rephrases. After each test case is annotated by three human annotators, we have 6k test cases with the annotator agreement.

Evaluation is performed on the NLU hypothesis level, using standard information retrieval metrics: Precision@N (P@N). The P@N measures if at least one rewrite in the first N candidates has a NLU hypothesis matched the y's.

8.3 Online Evaluation for Deployment

Ensuring the performance of a runtime system for general traffic has a great importance. We launch the UFS-QR (both global and personalized, separately) in production in an A/B testing setup and measure the defect rate decrease in live traffic.

9 RESULTS

9.1 Offline Evaluation

In this paper, performance is reported in terms of relative improvement over baseline[7]. Table 2 shows offline experimental results from global UFS-QR system. As described, DSSM based approach is shown as the baseline, whose performance will be reported as 0%. The DSSM baseline obtains an absolute p@10 around 40% for the test set. By introducing the retrieval layer with multi-model design in UFS-QR, we see a relative improvement of 15.1% at p@1

[7]Given the precision of baseline system, we calculate the relative precision performance changes and report this in %. It is calculated as $100 \times (p_c - p_b)/p_b - 100$, where p_b is the baseline precision, and p_c denotes any comparing system's precision.

Table 2: Summary of global UFS-QR experiment results.

	p@1	p@10
baseline DSSM	0%	0%
UFS-QR Retrieval CNN + DNN	+15.1%	+25.3%
+ UFS-QR Ranking Neural Model	+36.2%	N/A
+ UFS-QR Ranking GBDT	+142.2%	N/A

Table 3: Ranking layer performance given correct retrievals

	p@1
Ranking Neural Model	0%
Ranking Neural Model: with correct retrieval	+100.7%
Ranking GBDT	0%
Ranking GBDT: with correct retrieval	+166.6%

Table 4: UFS-QR performance on personalized test case

	p@1
global UFS-QR	0%
personalized UFS-QR	+64.1%

and 25.3% at p@10. By feeding the retrieved candidates into the ranking neural model, we see a relative improvement of 36.2%. Finally, when we use the full UFS-QR system, feeding the feature extracted by the ranking neural model into the GBDT model, we achieve a significant relative improvement on p@1 of 142.2%. This significant boosted performance from GBDT comes from the combination of pure semantic comparison captured by the ranking neural models and the feedback signal carried in the designed IR features. For example, for a request "what's the weather forecast for papa michigan", the correct rewrite from the index is in a syntactically distant form, "weather report for paw paw michigan". While retrieval layer retrieved this rewrite candidate as one of top 10 retrievals, other candidates such as "what's the weather forecast for portage michigan" obtained a higher score from the retrieval layer. Ranking layer, utilizing document level features including NLU hypothesis information, could successfully rank the correct rewrite to final rewrite. Further analysis showed that the following features are one of the most important features for the ranking performance: L2 neural features, weighted defect score, weighted query impression, n-gram defect, n-gram impression, TF-IDF.

Additionally, we investigate further into the ranking layer performance, by passing correct retrieval candidates to the ranking layer. Results are shown in Table 3. We take the ranking layer performance from Table 2 as our baseline, in order to showcase how much improvement the ranking layer can achieve given the perfect retrieval results. In this case, we see a relative improvement in p@1 of +100.7% and +166.6% for the ranking neural model and the GBDT respectively. This demonstrates the importance of improving on the p@n from the retrieval layer, which we leave as future work. In production system, we carefully tweak the trigger threshold within UFS-QR to avoid bad rewrites.

Table 4 summarizes p@1 performance on the personalized test set. We take global UFS-QR system as the baseline here, whose p@1

Table 5: Examples from global and personalizated UFS-QR

Query	Global rewrite from UFS-QR
Tooth or dare	Let's play truth or dare
Do I have any rain sounds	Play heavy rain sounds
New master TV	Mute master TV
Where is the nearest floor in decor	Where is the nearest floor and decor
Play hit or love it	Play hate it or love it
Query	**Personalized rewrite from UFS-QR**
Turn off the Tasha's bedroom	Turn of Natasha's bedroom
What's the temperature in Wilkeson	What's the temperature in Wilkerson
How's traffic to BW airport	How's traffic to DFW airport
Turn on cam	Turn on kim
Play my new five playlist	Play my new finds playlist

performance is reported as 0%. Note that the personalized test set mostly contains utterances from the extreme tail distribution (e.g. utterances that regard user's specific device setup). Global UFS-QR very often does not have any confident rewrites for such test cases. On personalized test cases, we observed a great difference in trigger rate, where personalized UFS-QR is triggered 12 times more often than global UFS-QR. Personalized UFS-QR improves the global UFS-QR on this test set by relative 64% on p@1, while improving the trigger rate by around 12 times. We observe that UFS-QR can recover personalized level information leveraging personalized index as well as semantic similarity measure, including user's device name, playlist name.

Table 5 shows rewrite examples from global and personalized UFS-QR. From global UFS-QR, we see rewrite patterns where user's slip of the tongue (e.g. "tooth or dare") is recovered into a better-formulated utterances. It is also noticeable that errors in entities such as song name and merchandise name are recovered. Also, some queries are rewritten to convey user's requests more clearly (e.g. rewriting a vague question into an actionable query "play heavy rain sounds"). From personalized UFS-QR, we see corrections that leverage individual user's previous usage. We often see personalized rewrites that recover from errors on user's device setup. For example, a query "turn on cam" can be a frequent request globally, as many users utilize a smart camera device. However, we see that this is an error for the user who often uses a device called "kim". Personalized UFS-QR could successfully recover from this error, rewriting the query to "turn on kim". Other rewrite patterns we see include recovering an error in a weather location for the user as well as locating a more relevant airport to the user based on user's previous usage.

9.2 Online Evaluation

After the offline experiments, we launched global UFS-QR in production in an A/B testing setup. We compared the performance of global UFS-QR system against no UFS-QR rewrites within English speaking users in the US, for a week in production. In the production, as a baseline, we had the QR system based on Ponnusamy et al. [18]. We observed significant[8] reduction of defect rate (13%). Launch of UFS-QR increased the number of global rewrite by 13.5%[8].

Launching personalized UFS-QR system on top of the global UFS-QR led to an additional significant defect rate reduction of 4%[8]. Compared to having only global UFS-QR, personalized UFS-QR significantly reduced user rephrases by 4.33%[8]. Moreover, launch of personalization layer further improved our sentiment metrics, leading to 1.46%[9] of decrease in dissatisfaction metric. The total number of rewrites by global UFS-QR and personalized UFS-QR are comparable, showing almost 1:1 ratio in production. Once global and personalized UFS-QR systems were fully launched in production, the total number of rewrites was increased by 46%.

As a part of production monitoring, we monitor the number of new, unseen rewrites in production. For example, a rewrite pair "play watch your hands" to "play wash your hands" can be a newly occurring rewrite pair, that has never been triggered in the production system earlier. We observe that UFS-QR systems introduce new, unseen rewrites significantly more often than the baseline system [18]. On a weekly basis, we observe that global UFS-QR introduces new, unseen rewrites 75% more than the baseline. Personalized UFS-QR introduces such rewrites 30% more than the baseline.

10 CONCLUSION

In this paper, we introduced a two stage retrieval/rank based query rewrite system UFS-QR for friction reduction for spoken dialogue system. We further extend the scope and introduce personalized UFS-QR, where we build personalized index and provide rewrites for personalized usage. We demonstrated the UFS-QR has the following advantage compared with other previously proposed systems: it requires no annotation data for model training by leveraging user interaction data; it provides a generalized rewrite solution without any constraints on the query text and incorporates users' interaction history and friction ratio into both index generation and offline model training; it reduces friction from all sources in the spoken dialogue system instead of focusing on one component (e.g. ASR or NLU). Our offline experimental results showed a significant improvement from UFS-QR over a neural semantic search model. Online evaluation showed significant friction reduction with a large scale rewrite traffic volume.

[8] p-value < 0.001

[9] p-value < 0.023

REFERENCES

[1] Praveen Kumar Bodigutla, Longshaokan Wang, Kate Ridgeway, Joshua Levy, Swanand Joshi, Alborz Geramifard, and Spyros Matsoukas. 2019. Domain-Independent turn-level Dialogue Quality Evaluation via User Satisfaction Estimation. *arXiv preprint arXiv:1908.07064* (2019).

[2] Daniele Bonadiman, Anjishnu Kumar, and Arpit Mittal. 2019. Large Scale Question Paraphrase Retrieval with Smoothed Deep Metric Learning. In *EMNLP 2019, W-NUT*.

[3] Chris J.C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An Overview. *MSR-TR-2010-82* (2010).

[4] Zheng Chen, Xing Fan, Yuan Ling, Lambert Mathias, and Chenlei Guo. 2020. Pre-Training for Query Rewriting in A Spoken Language Understanding System. In *ICASSP*.

[5] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *ICLR* (2020).

[6] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to attend, copy, and generate for session-based query suggestion. In *CIKM*. ACM, 1747–1756.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] Jinxi Guo, Tara N. Sainath, and Ron J. Weiss. 2019. A spelling correction model for end-to-end speech recognition. In *ICASSP*.

[9] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to rewrite queries. In *CIKM*. ACM, 1443–1452.

[10] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient Natural Language Response Suggestion for Smart Reply. (2017).

[11] Chao-Wei Huang and Yun-Nung Chen. 2020. Learning ASR-Robust Contextualized Embeddings for Spoken Language Understanding. In *ICASSP 2020*.

[12] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data. (2013).

[13] Johnson, Jeff, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* (2019).

[14] Yuan Ling, Benjamin Yao, Guneet Kohli, Tuan-Hung Pham, and Chenlei Guo. 2020. IQ-Net: A DNN Model for Estimating Interaction-level Dialogue Quality with Conversational Agents. In *Proceedings of KDD Workshop on Conversational Systems Towards Mainstream Adoption*.

[15] Deepak Muralidharan, Justine Kao, Xiao Yang, Lin Li, Lavanya Viswanathan, Mubarak Seyed Ibrahim, Kevin Luikens, Stephen Pulman, Ashish Garg, Atish Kothari, and Jason Williams. [n.d.]. Leveraging User Engagement Signals For Entity Labeling in a Virtual Assistant. In *NeurIPS 2018 Conversational AI Workshop*.

[16] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[17] Dookun Park, Hao Yuan, Dongmin Kim, Yinglei Zhang, Spyros Matsoukas, Young-Bum Kim, Ruhi Sarikaya, Chenlei Guo, Yuan Ling, Kevin Quinn, Tuan-Hung Pham, Benjamin Yao, and Sungjin Lee. 2020. Large-scale Hybrid Approach for Predicting User Satisfaction with Conversational Agents. In *NeurIPS 2020 Workshop on Human in the Loop Dialogue Systems*.

[18] Pragaash Ponnusamy, Alireza Roshan-Ghias, Chenlei Guo, and Ruhi Sarikaya. 2020. Feedback-Based Self-Learning in Large-Scale Conversational AI Agents. In *The Thirty-Second Annual Conference on Innovative Applications of Artificial Intelligence*.

[19] Stefan Riezler and Yi Liu. 2010. Query rewriting using monolingual statistical machine translation. *Computational Linguistics* 36, 3 (2010), 569–582.

[20] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.

[21] Alexander Schmitt and Stefan Ultes. 2015. Interaction quality: assessing the quality of ongoing spoken dialog interaction by experts—and how it relates to user satisfaction. *Speech Communication* 74 (2015), 12–36.

[22] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. 2014. A Convolutional Latent Semantic Model for Web Search. (2014).

[23] Prashanth Gurunath Shivakumar, Haoqi Li, Kevin Knight, and Panayiotis Georgiou. 2019. Learning from past mistakes: improving automatic speech recognition output via noisy-clean phrase context modeling. *APSIPA Transactions on Signal and Information Processing* 8 (2019).

[24] Milad Shokouhi, Rosie Jones, Umut Ozertem, Karthik Raghunathan, and Fernando Diaz. 2014. Mobile query reformulations. In *SIGIR*. ACM, 1011–1014.

[25] Milad Shokouhi, Umut Ozertem, and Nick Craswell. 2016. Did you say u2 or youtube?: Inferring implicit transcripts from voice search logs. In *WWW*. 1215–1224.