Personalized Search-based Query Rewrite System for Conversational AI

Eunah Cho **Ziyan Jiang** **Jie Hao** **Zheng Chen**
Saurabh Gupta **Xing Fan** **Chenlei Guo**

Amazon Alexa AI
{eunahch, ziyjiang, jieha, zgchen,
gsaur, fanxing, guochenl}@amazon.com

Abstract

Query rewrite (QR) is an emerging component in conversational AI systems, reducing user defect. User defect is caused by various reasons, such as errors in the spoken dialogue system, users' slips of the tongue or their abridged language. Many of the user defects stem from personalized factors, such as user's speech pattern, dialect, or preferences. In this work, we propose a personalized search-based QR framework, which focuses on automatic reduction of user defect. We build a personalized index for each user, which encompasses diverse affinity layers to reflect personal preferences for each user in the conversational AI. Our personalized QR system contains retrieval and ranking layers. Supported by user feedback based learning, training our models does not require hand-annotated data. Experiments on personalized test set showed that our personalized QR system is able to correct systematic and user errors by utilizing phonetic and semantic inputs.

1 Introduction

With the increased popularity of virtual assistant agents such as Alexa, Cortana and Siri, millions of users interact with spoken dialog system on daily basis. Some of the user interactions lead to *defect* experiences, where users do not get what they requested for, or the AI agent has to engage with the user again to clarify the user request. Such errors originate either from the spoken dialogue system itself (e.g. automatic speech recognition (ASR) error, natural language understanding (NLU) error, etc.) or the user. For example, an ASR error can lead to "play son in dance" instead of the correct query "play stolen dance". Often users' slips of the tongue can lead to the similar issue. It can be also the case that a partial song name is used in the request, or lyrics instead of the song name. The goal of QR component is to automatically recover from the defect and achieve better user experience.

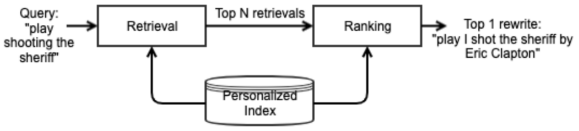

Figure 1: Search-based personalized rewrite framework

In production spoken dialogue systems, QR component is often triggered when the conversational AI cannot process user requests with a good confidence. For example, if ASR confidence score is low or named entity tagger (NER) confidence score for NLU is low, QR component can be triggered to automatically map a user query to another form, so that the dialog system can be more robust.

A crucial nature of QR is that often it needs to reflect personal preference or personalized error types to recover from the defect. While certain generic defect patterns may occur globally for many users, personalized error recovery is often expected to correctly fulfill each individual user's request. For example, a query "turn on the moon" might be a mis-recognized utterance requesting for the Moonlight Sonata. For other users who have a smart light device called "the moon lamp", this might be a request to turn on the lamp. Personalized QR can be leveraged for such scenario, providing rewrites such as "turn on the moonlight sonata" or "turn on the moon lamp".

Inspired by Fan et al. (2021), we approach the personalized QR task here with a search-based solution. A search system mainly comprises of two stacks operating sequentially: retrieval and ranking. Figure 1 shows the overall framework of our search-based personalized rewrite system. Personalized index contains successful utterances from each user. Given the user query, retrieval model first retrieves top N candidates from the user's personalized index. In the retrieval stage, we aim to obtain the most relevant rewrite candidates with a

Proceedings of the Third Workshop on Natural Language Processing for Conversational AI, pages 179–188
November 10, 2021.

good recall. The retrived candidates are ranked by the ranking model, utilizing personalized features extracted from personalized index. Top 1 ranked candidate is returned as a final rewrite for the query. In the example shown in Figure 1, we observe that the rewrite does not only correct the song name, but also finds the correct artist name based on the personalization for the user.

We utilize semi-supervised learning to obtain data and create the personalized index for each user in the conversational AI system. The personalized index primarily serves as a search pool for each user in the retrieval component. In our work, the index encompasses diverse affinity layers, which are utilized further as features in the ranking process. In Section 3, we describe how the documents are constructed for personalization index. In Section 4, we present the retrieval models we explored. Section 5 describes the ranker model and its diverse set of features.

2 Related Work

Many previous work focused on solving the QR problem in a non-personalized use case. In Ponnusamy et al. (2020), authors proposed a markov chain model as a collaborative filtering mechanism to mine users' reformulation patterns. Once the patterns are discovered, rewrite pairs are extracted (i.e. ASR recognition of defect query - detected reformulation interaction). At runtime, if a query is an exact text match with a defect query, a rewrite will be triggered using the offline mined pairs. Different from the Markov chain model, Chen et al. (2020) proposed a retrieval model for QR task, utilizing a query encoder that incorporates contextual language modeling pre-training.

At the same time, QR problem has been explored for other tasks as well. In Bonadiman et al. (2019), authors explored paraphrase retrieval for defect reduction on question answering task. A machine translation based method (Riezler and Liu, 2010) was explored for a QR problem in a search engine.

Fan et al. (2021) further extended previous work on search-based QR in spoken dialogue system to both global layer (serving non-personalized, generic rewrites) and personalized layer. They utilize the search based methods and metrics such as similarity measures. Compared to the markov chain model (Ponnusamy et al., 2020), this work provided a further flexibility to generalize over unseen input queries. For global layer, authors explore the importance of a ranking layer leveraging a neural feature extractor and a tree model. In Roshan-Ghias et al. (2020), authors present a retrieval model compared with a pointer-generator network with hierarchical attention to perform personalized rewrites within smart home domain. In our work, we focus on the search-based QR system that utilizes retrieval and ranking components. One strength of the search-based QR is that as the rewrite candidate pool is defined in the index, rewrite quality can be tightly controlled. Compared to generation-based methods, which may provide enhanced flexibility and diversity of the rewrites, this provides an advantage for system stability, especially for a task like QR which is strongly relevant with runtime production setup.

Our work distinguishes itself from previous work from the following aspects. First, we significantly extend the personalized index to incorporate diverse affinities and associated personal preferences. Our QR system is not limited to certain domains or usage, but capable of providing rewrites across all domains and functionalities within the spoken dialogue system (e.g. playing music, reading books, weather forecast, handling smart home devices, etc.). The retrieval component is further extended so that not only it utilizes the utterance information, but it incorporates phonetic information as well as ASR n-best information. While personalized search-based QR system in Fan et al. (2021) only leveraged a retrieval component, we introduce a ranking model with personalized features.

3 Personalized Index

In order to support personalized rewrites, we build an index for each user, leveraging individual interaction history. Personalized index reflects non-defect experiences for each user[1] observed within last 30 days of time window. We utilize a defect detection model such as Ling et al. (2020); Gupta et al. (2021) as well as rule-based criteria to obtain non-defect turns. The rule-based criteria utilizes both users' response (if user barged in or stopped agent's response, the turn is defect) as well as agent's response (if agent responds as "sorry, I do not know that" or similar, the turn is defect). To better define the search space and consider runtime aspects, we limit the personalized index size, by selecting top l history utterances based on utterance count and recency from the user. In order to control runtime

[1] All user information is in a de-identified format.

latency, we use $l = 100$ in this work.

Our personalized index contains different layers to represent fine-grained affinities; utterance (with NLU hypothesis), entity, intent, and template. The utterance layer directly serves as the search space for the personalized retrieval. The other layers provide personalized features utilized for ranking process. In the following sections, we describe each of them in detail.

3.1 Utterance Layer

As mentioned, personalized index is constructed for each user of the conversational AI system. Utterance layer can be considered as an entry point, where we trace non-defect utterances spoken by each user. Figure 2 shows an example personalized index.

The documents in the index are 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 dialog system, and can be represented in the following format "domain | intent | slot_type:slot_value". For example, given a query "put on cocomelon", we have an NLU hypothesis of "Music | PlayMusicIntent | ArtistName:cocomelon". The *domain* is the general topic of a query, e.g. "Music", "Weather". The *intent* reflects the action the user wants to take, e.g. "PlayRadio", "WeatherInfo"; The *slot-types/values* are results of entity labeling task. In the example, we can see that two queries spoken by the user "play cocomelon" and "put on cocomelon" are grouped together as they led to the same NLU hypothesis. For each NLU hypothesis and query, we trace how often the user has used this (*user count*) as well as how often this led to a defect experience (*user defect*).

3.2 Entity Layer

We further expand the affinity to include entity information so that the entity affinity information can be used during the ranking process. By *entity*, we refer to the slot values that we can observe in the NLU hypothesis. In the example index shown in Figure 2, we can extract the information that the user uses the entity "cocomelon" 30 times within the time window, and among them the defect experience was 3 times. To obtain an index with fine-grained affinity representation, we further gather the information on the joint occurrences between

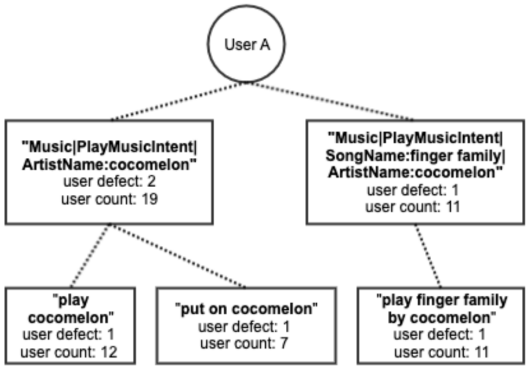

Figure 2: Example of utterance layer in personalized index

the slot types and the entity. Using this, we can reflect different usage pattern from each user. For example, for the users who frequently listen to "cocomelon" using an audio device, the entity would appear more frequently with ArtistName slot type. On the other hand, for the users who frequently watch videos of "cocomelon", the entity would appear more frequently as a VideoName enity. Such fine-grained usage pattern is leveraged as personalized features in the ranking layer (Section 5).

3.3 Intent Layer

Similarly to entity layer, we extract intent usage information. For each intent used by each user (e.g. "PlayMusic", "PlayVideo"), we trace user count as well as defect information.

3.4 Template Layer

By *template*, we refer to the utterance structure. For example, for a same-goal request, certain users may say "In my kitchen put John Lennon's song Imagine" while others may say "play Imagine by John Lennon on my kitchen device". We find those syntactic differences another way to represent user affinity and introduce the template layer. For template, entities are replaced to their slot-type and we only utilize the carrier phrase (e.g. "play SongName by ArtistName on my DeviceLocation device"). For each template, same as other affinity layers, we trace user count and user defect information.

4 Retrieval Component

In a search system, retrieval layer aims to retrieve a set of relevant documents from a great amount of documents. The goal is to retrieve them with good recall, with low latency and computational cost.

Following the work in Chen et al. (2020); Fan et al. (2021), we use a Deep Structured Semantic Models (DSSM) (Huang et al., 2013) based embedder. The learning objective is to project the embedding of input query (e.g. "turn on tape timer") and that of target rewrite (e.g. "turn on tea timer") closely.

4.1 Training Data Selection

For training data, as a baseline, we use weak-labeled data similar to Fan et al. (2021). We first find two consecutive user utterances, where the first turn was defect, but the second turn was successful. We utilized a DNN-based defect detection model such as Ling et al. (2020). In order to decrease potential noise in the data and find the pairs where the second utterance is indeed a rephrase of the first utterance, we further apply filters based on edit-distance and ASR n-best. For edit-distance based rephrases, we filter so that the utterance pairs have an edit distance smaller than d, and the time gap between the two utterances are shorter than t seconds. Additionally, we harvest ASR n-best-based rephrases. Two consecutive user utterances whose time gap is shorter than t seconds, if the first turn's ASR n-best ($n > 1$) is same as the following second turn's ASR 1-best, we consider them as rephrase as well. After preliminary experiments to control the trade-off between noise level and training data size, we chose $t = 45$ and $d = 5$. Maximum size for n is 5.

In order to decrease potential noise in above data, we apply another data selection method by adding a constraint on transitioning probability. From interaction history of each user, we calculate the probability of two utterances occurring one after another. We then apply a constraint so that $P(u_2|u_1) > p$, where u_1 denotes the first turn, and u_2 denotes the following turn. This helps to clean up the data further and filter out noise introduced in the training data for the personalized model. This way of data selection led to a smaller amount (60%) of rephrases as training data, compared to the baseline approach. In our experiments, we empirically set the $d = 7$ and $p = 0.5$.

4.2 Phonetic Information into Retrieval

We observe that user defects often stem from ASR error as well as users' slip of the tongue. For example, a query "did humidifier on" is an ASR error of "dehumidifier on". Another example includes an erroneous recognition "can you play alien bridges", which was rephrased by the user into "can you play leon bridges". While the baseline retrieval model (Fan et al., 2021) utilizes surface form representation of user utterances and their embedding space, we aim to further expand this to also incorporate phonetic information.

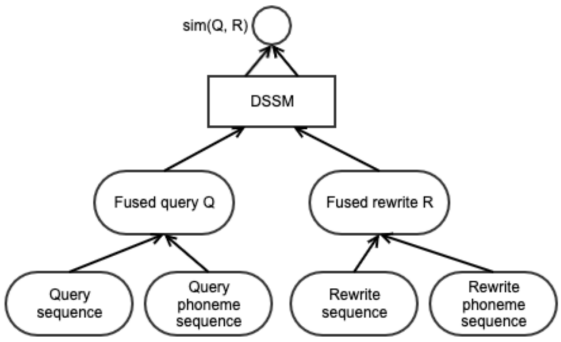

Figure 3: Illustration of DSSM model with phoneme sequence

In order to obtain the phoneme sequence of each utterance, we utilize a grapheme to phoneme (G2P) model based on the Transformer (Vaswani et al., 2017)[2]. Baseline DSSM model uses the textual representation of the query (e.g. "did humidifier on") and the target rewrite (e.g. "dehumidifier on"). While still utilizing the DSSM structure, we introduce another layer to fuse textual input representation and the phonetic sequence representation (e.g. "d I d h j u m I d @ f aI @‘ A n"), for both query and target rewrite. Standard G2P outputs phoneme sequence in their monophone sequence. Our preliminary experiment showed that augmenting word boundaries in phoneme sequences (e.g. "dId hjumId@faI@‘ An"), making the phoneme sequence more comparable to the utterance sequence length, leads to better performance. Figure 3 depicts how the phoneme sequence is incorporated into the DSSM model.

The neural encoder, thus, learns to capture latent syntactic, semantic and phonetic information for the query and the rewrite candidates. For the final similarity comparison, we use the scoring function shown in Eq. 1, where cos represents cosine distance.

$$S(x, y) = cos\,(\mathbf{h}, \mathbf{r}) \quad (1)$$

$\mathbf{h}$ is the embedding of input query x and $\mathbf{r}$ is the embedding of the rewrite candidate y, generated by passing them through the neural encoder respectively.

[2]We utilized Sockeye package (Hieber et al., 2017)

We calculate the score S given the input query and rewrite candidates (utterances) from personalized index. Top N candidates are chosen by their score S and passed down to the ranker component.

4.3 Utilizing ASR n-best

The QR system is triggered when the conversational AI cannot process user queries with a good confidence. Many previous designs for QR component, such as Fan et al. (2021); Chen et al. (2020), consume ASR 1-best as an input query to the QR system. Instead of consuming only ASR 1-best to our QR system, in this work, we expand our retrieval component to also consume the full ASR n-best list. By expanding the input query to a list of ASR hypotheses, we aim to boost the recall in the retrieval layer. For example, for a query (ASR 1-best) "turn on prayer lights", considering its ASR n-best such as "turn on foyer lights" in addition might be helpful to retrieve the correct rewrite "turn on party lights".

We expand the scoring function to consider input query's ASR n-best, x_0, x_1, ..., x_{n-1}. Rewrite candidates are retrieved based on the maximum score S obtained between them and any of the input ASR n-best. In order to consider runtime latency, we limit the n to 5.

4.4 Model Training

The neural encoder in the retrieval layer takes character-level trigram and word-level as input. We sum the embeddings of all tokens and pass through three layers of fully connected MLP with 512 hidden layer size. The final embedding size is 300. The experiments are performed using AllenNLP[3].

5 Ranking Model

Given retrieved rewrite candidates, ranking layer aims to rank the most desired rewrite to the top. Thus, precision is critical in the ranking layer. In this work, we build a gradient boosting model that utilizes both global level features as well as personalized features.

In our personalized QR system, we retrieve top 10 rewrite candidates in the retrieval component and pass them down to the ranking component. The top 1 rewrite whose ranking score is higher than an empirically chosen θ is selected as the final rewrite.

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

5.1 Global Features

By *global* features, we refer to non-personalized, generic features that we can consider for a rewrite ranking model. Even though the major use case of the model is for personalization, our intuition is that there are generic features that the model can learn to boost good rewrites (e.g. how similar query and rewrite utterances are).

Global features include the followings. **Global IR features** includes many of traditional information retrieval (IR) features, including TF-IDF, BM25 (Robertson et al., 1995), (weighted) number of queries, (weighted) number of tokens and n-gram probability. To extract global IR features, we utilize a document aggregated in global level, similarly to Fan et al. (2021). Per each NLU hypothesis, we aggregate relevant queries, their frequency and defect information. To obtain **embedding features**, we use two utterance encoders; one based on utterance string, another based on phonetic sequence. Each model is used to represent utterance strings (query and rewrite candidate) in their embedding space. We then obtain similarity metrics between the query and the rewrite candidate, such as cosine distance, Euclidean distance, Manhattan distance, etc. The utterance encoder follows the DNN-based baseline retrieval model (Huang et al., 2013; Fan et al., 2021). Phonetic sequence encoder follows the model described in Section 4.2 but CNN-based (Shen et al., 2014). For both encoders, we use three layers of fully connected MLP with 512 hidden layer size. The final embedding size for both DNN and CNN is 300.

To obtain **phonetic features**, we first represent query and rewrite candidates in their phonetic sequence. Similar as in Section 4.2, we use G2P model to translate the utterance sequence to their phonetic sequence. Their textual similarity is calculated in Jaccard distance, edit-distance, as well as BLEU (Papineni et al., 2002). In addition, we utilize **utterance popularity** information as one of the global features. For this, we look into the number of users who spoke the utterance in a time window (i.e. 30 days).

Finally, **defect features** trace defect metrics for query and rewrite candidate. In order to build defect metrics, we observe production traffic utterances within a certain time window (i.e. 30 days) and obtain average defect metrics per utterance. Defect metrics consists rule-based ones (e.g. average *termination*, representing how often users

follow up with "stop" or such after the query, average *barge-in*, representing how often users barged in right after they make a query, etc.) and model-based ones. Model-based ones include *rephrase* score from a rephrase detection model, indicating whether the query has been rephrased by the next turn. Rephrase detection model is a BERT (Devlin et al., 2018) based model, fine-tuned with human annotated rephrase pairs, and outputs the probability that the second query is a rephrase of the first query. Another model-based metric is *defect* score obtained from a DNN model for estimating dialogue quality (Ling et al., 2020).

5.2 Personalized Features

The ranker further utilizes rich personalized features. For **user affinity features**, we extensively used the affinity layer information from the personalized index (Section 3). For example, utilizing the entity layer in the personalized index, we extract user count and user defect information for an entity within the rewrite candidate. Intent and template layers are utilized in the similar manner. Intuition is that if the user hardly uses "PlayVideo" relevant intents but "PlayMusic" relevant intents more frequently with a certain entity (e.g. "cocomelon"), the ranker should be able to leverage this information to further promote the correct rewrite candidate. **User embedding** is obtained by utilizing personalized index described in Section 3, specifically utterance layer. We use the DNN based utterance encoder and obtain utterance embedding of 300 dimension for each utterance in user's personalized index. Each utterance embedding is weighted by user's utterance count and averaged. The averaged utterance embedding is used as the user's embedding. The intuition is that users' utterance usage will be reflected to the averaged embeddings. Similarly to the global features, we also construct **user IR features** by building a document for each user, extracting utterance frequency and defect information. Utilizing the document, we extracted the IR features comparable to the global IR features, such as TF-IDF, BM25, etc.

5.3 Training Data and Model Training

For model training, we obtained an adequate amount of rephrases from production traffic[4], following the methods described in Section 4. We run each rephrase through the personalized retrieval layer (Section 4) and obtain top 10 retrievals. If the retrieval matches rewrite label, it will serve as a positive label during the training. If not, the retrieval becomes a negative example.

In addition, we run the rephrases through a global retrieval model and obtain top 10 retrievals as well. Global index building follows the work in Fan et al. (2021), and we use the baseline retrieval model described in Section 4. By incorporating retrievals from both personalized and global layer, we aim to enhance model robustness. As personalized index reflects each user's preference and usage pattern, the retrievals are already rather distinctive for each other, leading to a relatively easier ranking problem. On the other hand, a global index contains documents aggregated from many users and can provide a relatively more challenging ranking problem. For example, given an input query "play shooting the sheriff", personalized index may have one strong rewrite candidate "play I shot the sheriff by Eric Clapton", reflecting user's previously shown preferences. However, global index can provide multiple relevant candidates, such as "play I shot the sheriff by Bob Marley" or "play sheriff". The global and personalized features together reach ∼730 dimensions of feature vector.

The ranker is trained using LightGBM (Ke et al., 2017) with DART (Vinayak and Gilad-Bachrach, 2015), where we set the max tree depth to be 20, learning rate to be 0.1, and the number of leaves to be 300.

Table 1: Retrieval model performance in precision and trigger rate. All performance is reported in relative improvement compared to the Baseline.

No.	System	Precision	Trigger Rt.
1.	Data Selection	+3.35%	+18.88%
2.	+ Phoneme	+6.56%	+55.94%
3.	+ ASR n-best	+6.32%	+119.85%

6 Experiments and Results

In order to build the test set, we first find 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 (i.e. 45 seconds) and their minimum edit distance is below an empirically chosen threshold, b) we can observe y in the user's personalized index with successful interaction indicated by the defect detection model. These

[4] all user data is processed to be de-identified.

Table 2: Top 1 retrieval from retrieval model. "Retrieval sys." denotes the systems in Table 1.

Query	Top 1 retrieved rewrite	Retrieval sys.
Play my first **hertz** playlist	Play my first **church** playlist	1, 2, 3
Play 104 **WKQR**	Play 104.1 **WTQR**	2, 3
What's the weather in **Wharton beach**	What's the weather in **Boynton beach**	2, 3
Turn on **huel**	Turn on **newel**	3

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.

6.1 Evaluation Metrics

Evaluation is performed on the NLU hypothesis level, in precision and trigger rate. The precision here measures how often the triggered top 1 rewrite's NLU hypothesis matches the correct rewrite y's NLU hypothesis. Since our test set represents personalization opportunity, we also measure trigger rate, the ratio between rewrite-triggered test cases and all test cases. The QR component is triggered when the prediction score of the top 1 rewrite score is above an empirically chosen threshold. Trigger rate represents how often we can trigger the rewrite given personalization opportunities.

In this paper, performance is reported in terms of relative improvement over baseline. For example, 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.

6.2 Results and Analysis

Table 1 shows the retrieval component performance on the personalized test set. To evaluate the performance of retrieval component, we take the top 1 retrieved candidate whose score exceeds a threshold θ, same as Fan et al. (2021). The threshold is carefully chosen empirically to avoid bad rewrites in production system. Thus, Table 1 represents the scenario where only the retrieval component is available for the personalized QR task.

For baseline, we utilize the utterance-based retrieval model following the weak-labelled training data selection described in Fan et al. (2021). As described, all performance is reported as relative performance against the baseline.

We can see that utilizing the utterance-based retrieval model with data selection based on transition probability (row 1 in Table 1) improves both precision and trigger rate, even though the model is trained with a smaller amount of training data. Further significant improvement is achieved utilizing phoneme sequence on top of utterance sequence (row 2), and ASR n-best extension of input queries (row 3). It is especially noticeable that the baseline retrieval system's trigger rate is relatively low for the personalized test set. With all advancements (row 3), we can significantly improve the overall triggering rate with improved precision.

Table 3: Ranking model performance. Baseline is best performant retrieval system's top 1 retrieval.

	Precision	Trigger Rt.
Ranker: global features	-1.17%	+7.39%
Ranker: all features	+7.60%	+19.10%

Table 4: Final top 1 rewrite examples.

Query	Final rewrite
Show me news room	Show me Indee's room
Burns office on	Marin's office on
Olana soundtrack	Play Moana soundtrack
Start my working	Start my work day
Lights sports show	Lights for show

Table 2 shows examples of the top 1 retrieval from the retrieval component. We clarify which retrieval system achieved the rewrite by showing the system number corresponding to Table 1. We observe that the advancements in the retrieval component (data selection, phoneme sequence, and ASR n-best) led to the retrieval of personalized rewrites with a good confidence for many test cases. We can recover errors in user's playlist (hertz to church), frequently used radio station 104 WKQR to 104.1 WTQR). Other examples show that the retrieval system can correctly retrieve user's weather location (Boynton beach, Florida) for a mis-recognized

location (Wharton beach, Australia).

Table 3 shows the ranking component performance. As Baseline, we took the best performing retrieval component configuration (last row system in Table 1). When utilizing ranker with only global features, we observe a slight degradation in precision, but a great improvement in trigger rate. Utilizing both global and personalized features, both precision and trigger rates are significantly improved. Examples of top 1 ranked rewrites are shown in Table 4. In this table, we show the examples where the retrieval component could not trigger a rewrite with a good confidence score, but the ranking component can trigger the final rewrite. In addition to correcting errors in user's smart device setups (e.g. room name, device name), it is also noticeable that the personalized QR corrects errors in users' routine setups (last two examples in Table 4). Many conversational AI systems support such routine phrase setups, where user can preset certain phrases (e.g. "start my work day") for specific actions from the agent (e.g. turning on lights and fans in user's office). During our analysis, we observe many of such corrections on routine phrases. The examples showcase that personalized QR system can enable AI agents to perform desired actions upon corrected utterances, rather than causing defect for the users (e.g. "Sorry, I don't know that. Can you repeat?").

We further looked into feature weight and analyzed the feature importance in the ranker model (See Appendix). We learned that 40% of user affinity features are ranked in the top 50 important features. Global features had shown importance as well. For example, 80% of defect features made into the top 50 important features out of all features, showing that the generic, non-personalized defect information is still required for the ranking process.

7 Conclusions

In this paper, we introduced a personalized search-based query rewrite system for conversational AI system. In order to fulfill personalization, the individualized index tracks each user's successful interaction history and is further expanded to reflect diverse affinities. Retrieval model leverages both utterance and phoneme information, and retrieves top 10 rewrite candidates. The candidates are ranked by the ranking model, utilizing highly personalized features based on the affinity layers as well as generic features for the query rewrite task. Experiments demonstrated that the search-based QR system with personalization advancements brings a significant improvement on both precision and trigger rate on the personalized rewrite test set.

For future work, we will focus on improving the model components further, including incorporation of richer prior information into the system (e.g. pre-trained contextual language modeling). Another future direction is to include richer contextual and personalized information into the retrieval process. The goal is to utilize contextual, cohort information and expand the QR experience to the functionalities that the user has not explored yet. Also, we will explore including external common knowledge into the ranking process.

References

Daniele Bonadiman, Anjishnu Kumar, and Arpit Mittal. 2019. Large scale question paraphrase retrieval with smoothed deep metric learning. In *EMNLP 2019, W-NUT*.

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*.

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*.

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)*.

Saurabh Gupta, Xing Fan, Derek Liu, Benjamin Yao, Yuan Ling, Kun Zhou, Tuan-Hung Pham, and Chenlei Guo. 2021. Robertaiq: An efficient framework for automatic interaction quality estimation of dialogue systems. *2nd International Workshop on Data-Efficient Machine Learning (DeMaL)*.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*.

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.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154.

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 the KDD 2020 Workshop on Conversational Systems Towards Mainstream Adoption co-located with the 26TH ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD 2020), Virtual Workshop, August 24, 2020*, volume 2666 of *CEUR Workshop Proceedings*. CEUR-WS.org.

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*, pages 311–318.

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*.

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

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Alireza Roshan-Ghias, Clint Solomon Mathialagan, Pragaash Ponnusamy, Lambert Mathias, and Chenlei Guo. 2020. Personalized query rewriting in conversational ai agents. *arXiv preprint arXiv:2011.04748*.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. 2014. A convolutional latent semantic model for web search.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Rashmi Korlakai Vinayak and Ran Gilad-Bachrach. 2015. Dart: Dropouts meet multiple additive regression trees. In *Artificial Intelligence and Statistics*, pages 489–497. PMLR.

A Appendix

Ranker model utilizes $\sim$ 730 dimensions of features. Among them, Table 5 lists the most important features, sorted by their feature weight.

Table 5: List of important ranker features, sorted by importance.

Category	Feature description
Global IR	Number of character changes between query and rewrite candidate
Global IR	Number of NLU slot types in rewrite candidate
Utterance popularity	Number of users who spoke query
Utterance popularity	Number of users who spoke rewrite candidate
Defect	Average defect score for rewrite candidate
Global IR	Character level BLEU score between query and rewrite candidate
Global IR	TF-IDF
User affinity	User count for the rewrite candidate
Phonetic	Edit-distance on phoneme sequence of query and rewrite candidate
User affinity	User count for the NLU slot type and entity combination
Global IR	Entity overlap between query and rewrite candidate
Defect	Global count for rewrite candidate
Defect	How often users barged-in for the rewrite candidate
User affinity	User count for the NLU intent for the rewrite candidate
Defect	How often users rephrased for the rewrite candidate
User IR	TF-IDF based on user document
Global IR	Unigram jaccard distance between query and rewrite candidate
User affinity	Template user count for the rewrite candidate
Phonetic	Character level BLEU score for phonetic sequences of query and rewrite candidate
Global IR	Bi-gram jaccard distance between query and rewrite candidate
Defect	How often the rewrite candidate was not actionable by the agent
Embedding	DNN-based Euclidean distance between query and rewrite candidate
Embedding	DNN-based Manhattan distance between query and rewrite candidate
Defect	How often users rephrased for the query
User affinity	Whether user's frequently used entity is in the query
Global IR	Bi-gram edit-distance between query and rewrite candidate
Defect	Average defect score for query
User affinity	User count for the entity in the rewrite candidate
User affinity	User count for the entity in the query
Global IR	BM25
User IR	Normalized defect score for the user utterance
User IR	BM25 based on user document