Context-Aware Ranking Approaches for Search-based Query Rewriting

Anonymous ACL submission

Abstract

Query rewriting (QR) is an increasingly important technique for reducing user friction in large-scale conversational AI agents. Recently, the search based query rewriting system has been proven effective and achieved promising results. It is a multi-stage system that consists of two components orderly: retrieval and ranking. Specifically, given a query, a dual-encoder model retrieves top N rewrite candidates. Then a Gradient Boosted Decision Trees (GBDT) re-ranks the candidates by considering semantic and information retrieval (IR) features. However, although there is still a debate for the effectiveness of the neural ranking model on traditional Learning-to-Rank (LTR) problems, the neural LTR models for the QR task have not been explored. To this end, we first explore preliminary ranking models, including both tree-based (e.g., LambdaMART) and neural-based (e.g., point-wise, list-wise) ranking models. Furthermore, we propose a context-aware ranking approach by integrating the dialog context information into the ranking models. Experimental results demonstrate that the proposed context-aware ranking model outperforms the baselines significantly.

1 Introduction

Large-scale conversational AI agents such as Alexa, Siri, and Google Assistant, are becoming increasingly popular in real-world applications to assist users in daily life. It is unavoidable that the interactions involve *friction*. In general, there are mainly two types of errors that result in friction. First is the system error. Especially, Automatic Speech Recognition (ASR) might misrecognize the users' query, and Natural Language Understanding (NLU) misinterprets the semantics due to ambiguity or errors that come from the other components. For example, the ASR can lead to a wrong recognition of "voice room light off" instead of the user's intended "boys room light off". The second type is user ambiguity, such as a user's slip of the tongue or using abridged language while speaking the query.

Query rewriting (QR) (Grbovic et al., 2015; Chen et al., 2020; Yuan et al., 2021; Wang et al., 2021) aims to seamlessly rewrite the user's utterance in order to remove the potential friction. Currently, search-based query rewriting system (Fan et al., 2021; Cho et al., 2021) has been proven effective and widely used in practice to perform query rewriting. In general, there are two phrases in the search-based query rewriting system. First, given a new user query, the system retrieves top N candidate rewrites from a candidate set (i.e., index) using a siamese style encoder retrieval model. Then, a ranking model based on Gradient Boosted Decision Trees (GBDT) re-ranks the retrievals while considering semantic and information retrieval (IR) features and selects the top 1 in the ranking list as the final rewrite.

For the Learning-to-Rank (LTR) problem, previous efforts (Qin et al., 2021; Burges et al., 2005, 2006; Burges, 2010) for developing ranking models unusually only applies to numeric features and LTR datasets. The understanding of the effectiveness of these models is limited in the query rewriting scenario. Moreover, there has been a debate on the superiority between the tree-based ranking model and the neural ranking model (Qin et al., 2021; Li et al., 2019; Bruch et al., 2019), making it difficult for practitioners to choose ranking models. Last but not least, context information has been proven effective in boosting the model in many NLP tasks (Wang et al., 2017; Wu et al., 2018). However, few QR works, especially search-based approaches, consider the dialog context information in ranking.

Motivated by these threads of thought, we first investigate different types of ranking models in this work. Then, we develop an advanced context-aware ranking model that enhances the semantic features to promote the query rewriting performance by incorporating the dialog context. Our

contributions are two-fold. First, we propose preliminary ranking model structures for QR tasks, including tree-based and neural ranking models. The preliminary model structures can serve as foundations for building complex models for query rewriting. The proposed tree-based ranking integrates LambdaMART (Ke et al., 2017) which is the state-of-the-art gradient boosted decision tree model for ranking that uses pair-wise loss. For the neural ranking model, we explore both the list-wise ranking approach and the point-wise ranking approach. Second, we propose a context-aware neural ranking model that incorporates the context information. Experimental results demonstrate that the proposed context-aware ranking model significantly improves the performance.

2 Related Work

Query Rewriting In order to reduce users' friction and satisfy their daily demands, seamlessly replacing the user's request (i.e., query rewriting) for conversational agents has been paid more and more attention by researchers. Fan et al. (2021) and Cho et al. (2021) propose to leverage the search-based model, which consists of a DSSM based retrieval layer and a tree ranking layer, to handle global and personalized query rewriting. Su et al. (2019) use generation-based approaches to tackle the coreference and omission-specific scenarios. Besides them, Absorbing Markov Chain (AMC) (Ponnusamy et al., 2020) as a collaborative filtering mechanism is another attempt to mine the rephrase patterns and perform query rewriting. Search-based QR approaches have been successful, but past work is still stuck at using a tree-based model without considering dialog context information and other neural-based rankers at the ranking layer. Our work is to explore the various powerful ranking models and thus close this gap.

Learning to Rank Traditional Learning to rank (LTR) approaches focus on the problem where there are only numeric features and human ratings available (Qin et al., 2021). Some works (Nogueira and Cho, 2019; Cheng et al., 2016; Qin et al., 2020) on document matching and recommendation have leveraged neural components such as word2vec and BERT to generate numeric representations. Such a neural component provides advantages such as semantic modeling of highly sparse input. In this case, tree-based methods become less relevant due to their limitation in handling sparse features. The pioneering neural LTR models are RankNet (Burges et al., 2005) and LambdaRank (Burges et al., 2006). They apply feed-forward networks to the dense features to calculate the scoring functions and use implicit cost functions to handle the undefined derivatives issues in the neural LTR models. Later, tree-based LTR model such as LambdaMART (Burges, 2010) has proven to be more successful for solving real-world ranking problems. Recently, there are works exploring new model architectures (Pang et al., 2020; Qin et al., 2020), differentiable losses (Bruch et al., 2019), and leveraging more auxiliary information (Ai et al., 2018).

3 Approach

In this section, we first define the notations and formulate the problem. Next, we propose the various ranking models for QR, including the tree-based ranking model and neural-based ranking model. We then describe our advanced models, i.e., context-aware neural ranker that incorporates the context information associated with each query.

3.1 Notations and Problem Definition

Let q denote a specific query and $\mathbf{p} \in \mathcal{P}^m$, i.e., $\mathbf{p} = \{p^j\}_{j=1}^m$, denotes the m candidates associated with query q. Let $\mathbf{y} \in \mathcal{Y}^m$ be the ranking labels corresponding to query q and candidate list $\mathbf{p}$. Let $\mathcal{D}$ denote the an unknown but fixed joint probability distribution of q, $\mathbf{p}$ and $\mathbf{y}$. The ultimate goal of the learning problem is to learn a ranking function $f : \mathcal{P}^m \rightarrow \mathcal{Y}^m$ that can minimize the population risk, i.e.,

$$R(f) = \mathbb{E}_{(q,\mathbf{p},\mathbf{y})\sim\mathcal{D}} \left[\ell(f(q,\mathbf{p}),\mathbf{y})\right], \qquad (1)$$

where ℓ is a certain loss function associated to the learning problem. Since the underlying distribution of the query and the list of candidates and labels is unknown, one cannot directly minimize the population risk defined in (1). In practice, we are given a set of i.i.d samples $S = \{q_i, \mathbf{p}_i, \mathbf{y}_i\}_{i=1}^N \sim \mathcal{D}^N$. Thus, we can minimize the empirical risk, i.e.,

$$R_S(f) = \frac{1}{N}\sum_{i=1}^{N} \left[\ell(f(q_i,\mathbf{p}_i),\mathbf{y}_i)\right], \qquad (2)$$

to find an approximate minimizer.

In this work, we specifically consider the ranking function f as a function/model that scores and sorts the items in a list, i.e., given a query q and a list of m candidates $\mathbf{p} = \{p^j\}_{j=1}^m$, the goal is to find a

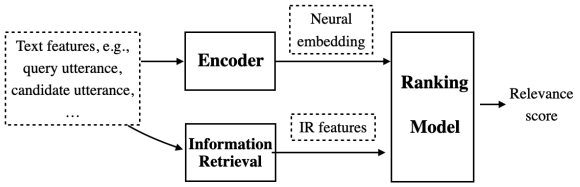

Figure 1: A brief description of the ranking layer framework.

parameterized ranking model f that returns a list of m ranking scores, i.e., $f(q, \mathbf{p}) \in \mathbb{R}^m$. Note that the list of ranking scores $f(q, \mathbf{p})$ can be also obtained in a candidate-independent way, i.e., given query q and candidate p^j for any $j \in [m]$, the model outputs a relevance score $f(q, p^j) \in \mathbb{R}$.

3.2 Preliminary ranking models

The ranking model f takes query q, top m candidates $\{p^j\}_{j=1}^m$ from the retrieval layer and output m ranking scores corresponding to the m candidates, i.e., $f(q, \mathbf{p}) \in \mathbb{R}^m$. In this work, the input data corresponding to query q, candidate p^j is composed of query utterance and candidate utterance. For example, a query utterance is "play kids you by coldplay" and the candidate utterance is "play fix you by coldplay".

Figure 1 provides the framework of our ranking model that mainly has three components: an encoder that maps utterance to neural embedding, an information retrieval layer that generates information retrieval (IR) features, and a ranking model that takes neural embedding and IR features and output ranking score. Below we introduce the details of each component, followed by the tree-based ranking models and neural-based ranking models.

Encoder Given the input data, i.e., query utterance and candidate utterance, we use the state-of-the-art encoder, i.e., BERT (Devlin et al., 2018) to generate neural embedding. Figure 2 (right) illustrates the BERT model applied to our encoding task. The input of the BERT encoder is a pair of query utterance and candidate utterance. Then the query utterance and candidate utterance is mapped to tokens with special token [SEP] indicating the split of query and candidate. We train the BERT model by simply modeling the ranking as a binary classification task and minimizing a cross entropy loss function which defined as follows. To be specific, among m candidates $\{p_i^j\}_{j=1}^m$ for a given query q_i, let $p_i^\star$ be the positive candidate, let $s(q_i, p_i^j) \in (0, 1]$ denote the relevance score, we minimize the binary cross-entropy loss

$$\sum_{i=1}^{N} \left\{ \log(s(q_i, p_i^\star)) - \sum_{p_i^j \neq p_i^\star} \log(s(q_i, p_i^j)) \right\} . \tag{3}$$

Note that the ultimate goal of the BERT model is to provide a neural embedding $\mathbf{e} \in \mathbb{R}^{768}$ (as shown in Figure 2) that represents the query and candidate pair. In the binary classification task, the relevance score $s(q_i, p_i^j)$ can be viewed as a single layer neural network with sigmoid activation function that takes the embedding $\mathbf{e}_i^j \in \mathbb{R}^{768}$ as input, where the index i, j correspond to the query and candidate pair (q_i, p_i^j).

IR Feature Besides the embedding generated from the BERT model, we also extract a group of conventional IR features following Fan et al. (2021). 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 (Papineni et al., 2002). Document features provide information at a document level 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. Overall, we generate a 243-dimensional IR feature vector, i.e., $\mathbf{r}_i^j \in \mathbb{R}^{243}$ for each query and candidate pair (q_i, p_i^j). Later we concatenate the neural embedding $\mathbf{e}_i^j$ and IR feature $\mathbf{r}_i^j$ and input them into the ranking model. Specifically, we have explored the tree-based ranking model, i.e., LambdaMART and neural ranking model.

Tree-based ranking model We consider Gradient Boosted Decision Tree (GBDT) as the ranking model. In contrast to the GBDT trained with point-wise binary logistic loss in Fan et al. (2021), we consider LambdaMATR (Wu et al., 2010; Burges, 2010) in this work. Different from general GBDT, during each boosting step, the loss of LambdaMATR is dynamically adjusted based on the ranking metric in consideration, i.e., the absolute difference between the NDCG values when two candidates s and j swap their positions in the ranked list. LambdaMART uses a pairwise logistic loss and adapts the loss by re-weighting each item pair using such absolute difference between

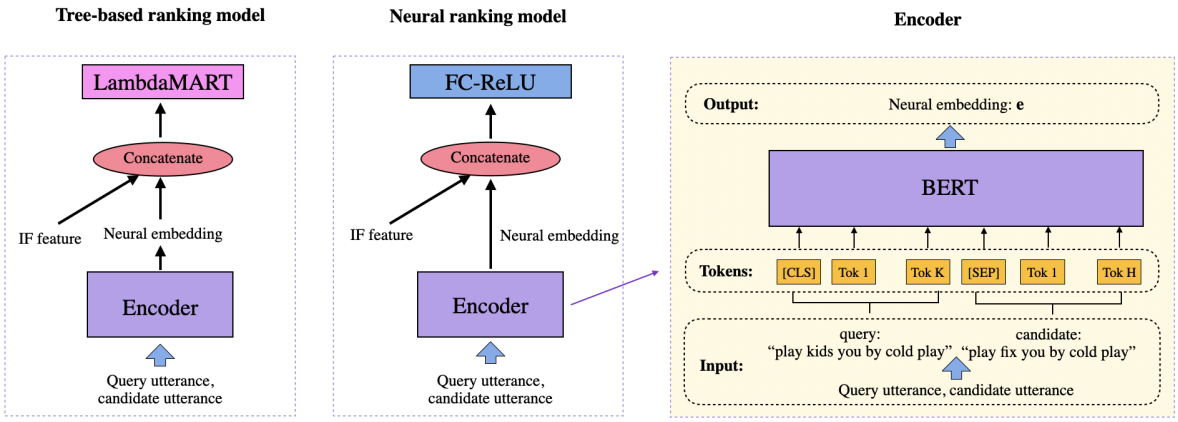

Figure 2: Structures of the tree-based ranking model (left), neural ranking model (mid), and encoder (right). For the neural ranking model, we can jointly train and fine-tune fully connected (FC) layers with the encoder. But for the tree-based ranking model, we can only first use a pre-trained encoder to generate neural embedding, then train the LambdaMART separately. We specifically consider BERT encoder, where [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token for separating query and candidate.

the NDCG when swapping the position of the candidate pair. Figure 2 (left) outlines the structure of the tree-based ranking model.

There are two popular public implementations of LambdaMART, namely λMART_{GBM} and $\lambda\text{MART}_{\text{RankLib}}$. λMART_{GBM} is more recent than $\lambda\text{MART}_{\text{RankLib}}$ and has more advanced features by leveraging novel data sampling and feature bundling techniques (Ke et al., 2017). Based on the finding in Qin et al. (2021) that λMART_{GBM} substantially outperforms $\lambda\text{MART}_{\text{RankLib}}$, we implement LambdaMART using the λMART_{GBM} library in this paper.

Neural ranking model The neural ranking model shares the same structure as presented in Figure 1 where the ranking model is specifically constructed by a fully connected neural network with ReLU activation function. Figure 2 (mid) gives the structure of the neural ranking model. The input of this model is also the concatenation of neural embedding $\mathbf{e}$ from BERT and IR feature vector $\mathbf{r}$ from the information retrieval layer. We highlight three major differences in the neural ranking model compared with the tree-based ranking model.

First, we implement the fully connected neural network with BERT as a unified model, which allows us to jointly train/fine-tune the ranking model and BERT encoder. The parameters of BERT are initialized using those of the pre-trained BERT as described in the tree-based ranking model. However, it is not feasible to jointly tree the LambdaMART and BERT for the tree-based ranking model. Second, we consider different loss functions for the neural ranking model, i.e., list-wise loss and point-wise loss, different from the pair-wise loss used in LambdaMART. Based on the loss function, we define two types of neural ranking models, i.e., list-wise model and point-wise model. Specifically, the loss function of the list-wise model is defined as follows: Given request q with m candidates $\{p^j\}_{j=1}^m$, list-wise model uses loss function

$$\ell\left(q, \{p^j\}_j\right) = -\log \frac{f\left(q, p^\star\right)}{\sum_{j=1}^m f\left(q, p^j\right)}, \quad (4)$$

where $p^\star$ is the positive candidate and $f\left(q, p^j\right)$ is the ranking score function for request and candidate pair (q, p^j). The point-wise model uses point-wise loss function:

$$\ell\left(q, \{p^j\}_j\right) = -\sum_{j=1}^m \log f\left(q, p^j\right), \quad (5)$$

where $f\left(q, p^j\right)$ is the ranking score function for request and candidate pair (q, p^j). Note that the difference between point-wise model and list-wise model is that the point-wise model treats each query candidate pair as independent sample in the training process. Whereas, the list-wise model considers the interplay between the candidates for each query as the training algorithm will boost $f\left(q, p^\star\right)$ for positive candidate and penalize $f\left(q, p^j\right)$ for negative candidates.

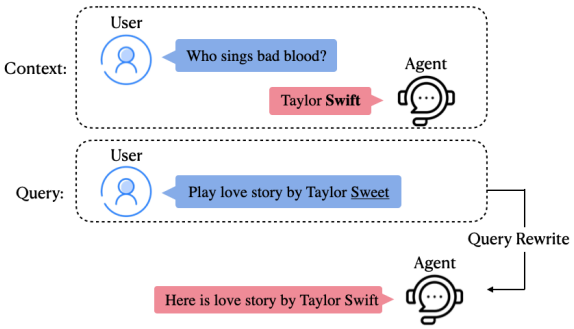

Figure 3: Example of the context in a dialogue session. In this example the query is "Play love story by Taylor Sweet". The context is the interaction between the user and agent before this query, i.e., "Who sings bad blood?", "Taylor Swift".

4 Context-Aware Ranking Model

This section introduces the context-aware neural ranking model. Inspired by Wang et al. (2021) that propose to use BERT to detect the rephrase within a multi-turn dialog context for query rewriting, we utilize the contextual information in the dialogue interaction between the agent and users in the ranking model. Unlike the Wang et al. (2021) taking the whole dialog session into the model to mine the rephrase, when we rewrite a query in real-time, only the context before the query is available, as shown in Figure 3. In our context-aware neural ranking model, we flatten the dialogue context before the query into one sequence and append the query to it. Then, we feed them to a pre-trained BERT to compute the embedding for context, query, and rewrite candidate. We introduce two special tokens: "[USER]" and "[AGENT]", which are used to prefix the user query and agent response in the contextual input, respectively. Take the Figure 3 as an example. After processing, the context and query are combined in the form of "[USER] Who sings bad blood [AGENT] Taylor Swift [USER] Play love story by Taylor Sweet." The context-aware neural ranking model shares the same structure as the neural ranking model as described in Section 3.2. In the context-aware ranking model, we replace the query with contextual input in the BERT encoder to generate neural embedding as shown in Figure 2.

5 Experiments

We evaluate the proposed approaches for solving the ranking problem in the query rewriting system. In particular, we conduct two sets of experiments. First, we compare the performance of the our preliminary ranking models, i.e., tree-based neural ranking model, neural ranking model, with the baseline *Data Augmented Self Attentive Latent Cross* (DASALC) proposed in (Qin et al., 2021), which gives the state-of-the-art ranking performance on LTR tasks. This set of experiment does not consider the context information. Second, in order to validate the proposed context-aware neural ranking model, we compare the proposed context-aware model with the preliminary models. We adopt the multi-model architecture and corresponding index for retrieval layer described in Fan et al. (2021) in our experiments.

Query	do the monster bash song
Candidates	play the song monster bash, label: 0
	play the monster mash song, label: 1
	the monster mash song, label: 0
	the monster bash, label: 0
	play the monster bash | label: 0

Table 1: An example of the data for experiments of preliminary ranking models.

5.1 Preliminary ranking models

Datasets We use the weak-labeled data similar to Fan et al. (2021). For the retrieval phrase, from de-identified historical interactions of a large-scale conversational AI agent, we find two consecutive user utterances, where the first turn was defected and the second turn was successful by utilizing a defect detection model Gupta et al. (2021). For ranking, each query has 5 rewrite candidates retrieved from an index with FAISS search (Johnson et al., 2017). For each query, the 5 candidates are also labeled as *positive* (label = 1) and *negative* (label = 0), i.e., positive means the candidate is a correct rewrite and negative means the opposite. For this dataset, each query has *at least one positive* candidate among the 5 candidates, i.e., correct rewrite. We collect 1-month period data for training and subsequent 1-week period data for validation and testing. Moreover, after getting the raw test set, we have manually identified the true labeled cases and removed the noise. More details of this dataset are deferred to the Appendix A.1.

Models and Baseline We evaluate the proposed tree-based and neural-based ranking mod-

els and the baselines: UFS-QR (Fan et al., 2021), DASALC (Qin et al., 2021). The DASALC model is a ranking model that combines data augmentation (DA), self-attention (SA) and Latent Cross (LC). For the proposed tree-based model, we implement the structure described in Figure 1 (left) with a pre-trained encoder. Since we specifically implement LambdaMART, we refer this model as LambdaMART. For the neural ranking model, we consider two types of neural ranking model: 1) Point-Wise Neural Ranking model (PWNR) with structure outlined in Figure 1 (mid) and point-wise loss function defined in (5); 2) List-Wise Neural Ranking model (LWNR) with the same structure in Figure 1 (mid), but list-wise loss function defined in (4). Thus, we outline the models below:

- **UFS-QR**: GBDT-based tree ranker proposed by Fan et al. (2021).
- **DASALC**: Data Augmented Self Attentive Latent Cross ranking model (Qin et al., 2021).
- **LambdaMART**: Tree-based ranking model with structure described in Figure 1 (left).
- **PWNR**: Neural Ranking model with structure described in Figure 1 (mid) and point-wise loss defined in (5).
- **LWNR**: Neural Ranking model with structure described in Figure 1 (mid) and list-wise loss defined in (4).

Training and Hyper-parameter Setting For the fair comparison, we implement the dssm-based retriever in Fan et al. (2021). Thus, all the ranking candidates are from the same retrieval model. Recall that we have three parts in our tree-based model, i.e., BERT encoder, Information Retrieval (IR) layer, and Ranking layer. Thus, we need to setup each part to obtain the final model. For the IR layer, we follow the setting in (Fan et al., 2021) to generates IR features for each query and candidate pair. For the BERT encoder in both tree-based ranking model (LambdaMART) and neural ranking model (LWNR and PWNR), we use a pre-trained BERT which is trained using the machine-annotated training data with objective loss function defined in (3). Then, we concatenate the IR feature from the IR layer and the neural embedding from the pre-trained BERT to form a new feature vector for each sample (query and candidate pair). Given the concatenated feature, we train the LambdaMART. Specifically, we implement LambdaMART using the λMART_{GBM} library. For λMART_{GBM}, following Qin et al. (2021), we do a grid search for number of trees $\in \{300, 500, 1000\}$, number of leaves $\in \{200, 500, 1000\}$, and learning rate $\in \{0.01, 0.05, 0.1, 0.5\}$.

Approach	Precision %
USF-QR	94.20
DASALC	97.65
LambdaMART	**98.22**
PWNR	98.05
LWNR	97.13

Table 2: Precision on test set at 10% trigger rate. The tree-based ranking model LambdaMART achieves highest precision among all models. The PWNR obtains competitive result comparing with other neural ranking models.

For the PWNR and LWNR, we jointly train the BERT encoder and the fully connected ranking layer. The BERT encoder is initialized using the weight of the pre-trained BERT. Using the pre-trained BERT encoder helps the encoder adapts to the domain of the machine-annotated data. We use point-wise loss defined in (5) as our loss function in the joint training phrase for PWNR, and list-wise loss defined in (4) for LWNR. We use ADAM as the optimizer for training all the models. The other hyperparameters of ADAM are default. For the learning rate, we follow the grid search method with linear search space $\{10^{-6}, 5 \times 10^{-6}, 10^{-5}, 5 \times 10^{-5}, 10^{-4}\}$. We use a fixed budget on the number of epochs, i.e., 5 to train the model and pick the model that performs best on the validation set.

Evaluation Metrics We use *trigger rate* and *precision* as the evaluation metrics. In practice, the query rewriting system will not rewrite/trigger every query from the users to consider the cases such that the query itself may not be defected or the candidate list does not consider the correct rewrite. Thus, the ranking model will only trigger the query with a certain confidence. In detail, the ranking model obtains a list of ranking scores corresponding to the list of candidates for each query. If the largest ranking score in the list exceeds a certain trigger threshold τ, we trigger the query and use the top 1 candidate as the rewrite. Then the trigger label for this query is 1. Otherwise, the trigger label is 0. The trigger rate denotes the ratio between

Approach	All		Has-Rewrite	
	TR %	Precision %	TR %	Precision %
USF-QR	17.67	74.91	29.26	89.89
LambdaMART	19.13	84.80	34.38	93.93
PWNR	16.79	73.56	27.75	88.91
CANR	19.12	87.60	35.64	94.80

Table 3: Trigger rate and precision of the proposed models on the Full-Scale test set. With the help of context, the context-aware neural model CANR obtains highest precision, even though it has a higher trigger rate.

Context	Candidate List and Label
USER: play me on bridges please AGENT: Did you mean play the song That Old Forth Bridge and Me? USER: no play leon bridges AGENT: I think you are asking for the song That Old Forth Bridge and Me, is that right? USER: no leon bridges **(Query)**	play leon bridges, label: 1 leon bridges, label: 0 play leon bridges radio no, label: 0 leon bridges live, label: 0 beyond leon bridges, label: 0

Table 4: An example of Full-Scale data.

triggered cases and all cases. The precision measures how often the triggered top 1 rewrite matches the correct rewrite. Mathematical definition of the trigger rate and precision is in Appendix A.1.

Experimental Results We present the results in Table 2. We mainly focus on comparing the precision of the proposed models with the baseline. The higher trigger rate means that more queries are triggered by the model. For a fixed trigger rate, the higher precision means that, among the triggered queries the more defected-query receive the correct rewrite by the model. Note that, naturally one can control the precision by manually adjusting the trigger rate. For example, one can increases the trigger threshold τ to trigger queries with high ranking scores, thus the precision can be increased since the analyst only account the precision for the queries with high ranking score (high ranking confidence). To offer a fair comparison, we set different values of τ for different models where the value τ of each model is decided by ensuring the model achieves the same trigger rate, i.e., 10% trigger rate on the test set. The result shows that The tree-based ranking model LambdaMART achieves highest precision among all models. The PWNR obtains competitive result comparing with other neural ranking models.

5.2 Context-aware neural ranking model

In this section, we conduct experiment to evaluate the proposed context-aware neural ranking model. We mainly focus on comparing the context-aware neural ranking model with LambdaMART and PWNR in this set of experiment since these two models achieves the best results among the prelimary models and baseline. We refer the Context-Aware Neural Ranking model to as CANR [1] in this section for simplicity.

Datasets In this set of experiment, we consider a more realistic dataset that includes all types of real-word queries. We refer to this dataset as Full-Scale Data. To be specific, in addition to the type of queries in preliminary ranking models' experiments, here we consider 1) queries that have context information and 2) queries that do not have a positive candidate retrieved by the retrieval model. Each sample is comprised of a query, its dialogue context between the user and the conversational AI agent, and 5 candidates for rewrite. Table 4 provide an example of this full-scale dataset with labels. In practice, for some queries, there is no correct rewrite among the 5 candidates provided by the retrieval model. For those queries without correct

[1] We use point-wise loss to train CANR since the point-wise model achieves better performance than the list-wise model as shown in Table 2.

Approach	All				Has-Rewrite			
	w/ context		w/o context		w/ context		w/o context	
	TR %	Precision	TR %	Precision %	TR %	Precision %	TR %	Precision %
PWNR	17.63	75.67	16.36	73.00	28.37	89.47	27.38	88.59
CANR	21.95	88.38	17.67	87.12	39.67	95.21	33.42	94.54

Table 5: Trigger rate and precision on Full-Scale test set with two subsets, i.e., **w/ context** and **w/o context**. For both queries with context and queries without context, the context-aware model achieves higher precision even with higher trigger rate. Thus, the context information can boost the performance of the neural model.

rewrite in the candidate list, it is hard to qualify the performance of the ranking model due to the inherent data constraint. Thus, we specifically consider a subset of the original dataset named **Has-Rewrite**, i.e., the set of queries that has *at least one positive* candidate in the candidate list. For each query, the 5 candidates are also labeled as *positive* (label = 1) and *negative* (label = 0), i.e., positive means the candidate is a correct rewrite and negative means the opposite. We provide details about this dataset in the Appendix A.1.

Main Result We present the results in Table 3. We mainly focus on comparing the CANR with LamdaMART and PWNR since LamdaMART and PWNR outperforms the other preliminary models in Section 5.1. To offer a fair comparison, we also set different τ for different models as in Section 5.1, where the value τ is decided by ensuring the model achieves the same trigger rate, i.e., 10% trigger rate on a validation set.

First, the result in Table 3 shows that the tree-based ranking model LambdaMART achieves higher precision compared to the neural ranking model PWNR, even with a higher trigger rate. Not surprisingly, the tree-based ranking model outperforms the neural ranking model as observed in Qin et al. (2021). However, with the help of context information, the neural model can obtain higher precision than the tree-based model, i.e., the context-aware neural ranking model achieves the highest precision, even though it triggers more queries. Note that the **ALL** Machine Annotated data also contains the queries that do not have a correct rewrite in the candidate list, meaning that for a fraction of triggered queries, the models always find false positive candidates. In consequence, the precision on the Full-Scale data is lower than the precision in Table 2, i.e., results on a less challenging dataset. Thus, we also report the trigger rate and precision on the **Has-Rewrite** subset, and the result in 5 shows the same story holds, i.e., the CANR model outperforms the tree-based ranking model and neural ranking model. The above result demonstrates that the context information can boost the model performance.

Effect of Context To further validate the observation that the neural model benefits from the context information, we also report the results for queries with context and queries without context. In particular, we split both **ALL** and **Has-Rewrite** into two subsets, i.e., **w/ context** and **w/o context**. "w/o context" denotes the current query is the first turn of a dialogue session and thus no available dialogue context. While "w/o context" denotes the current query is not the first turn in a dialog. The **ALL** dataset is composed of 33.78% queries with context and 66.22% queries without context. The **Has-Rewrite** subset is composed of 35.53% queries with context and 64.48% context without contex. The trigger rate and precision for the neural ranking model and context-aware neural ranking model are reported in Table 5. The result shows that for both queries with context and queries without context, the context-aware model achieves higher precision even with a higher trigger rate. Thus, the context information can boost the performance of the neural model. We provide case study in Appendix to further demonstrate the advantage of context.

6 Conclusion

We investigate the ranking approaches for search-based query rewriting system in this work. we first propose both tree-based and neural-based ranking models for QR. Then, we demonstrate the necessity of incorporating dialog context information into the Qr task and propose simple but effective context-aware ranking approach. We evaluate the proposed models on various data sets and demonstrate that the proposed context-aware ranking model can significantly improve the QR performance. Extensive analyses reveal that context-aware ranking model is robust on different scenarios even the query is the first turn in a dialog and no context available in testing stage.

References

Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. 2018. Learning a deep listwise context model for ranking refinement. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 135–144.

Sebastian Bruch, Masrour Zoghi, Michael Bendersky, and Marc Najork. 2019. Revisiting approximate metric optimization in the age of deep neural networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1241–1244.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96.

Christopher Burges, Robert Ragno, and Quoc Le. 2006. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems*, 19.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Zheng Chen, Xing Fan, and Yuan Ling. 2020. 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)*, pages 7969–7973. IEEE.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10.

Eunah Cho, Ziyan Jiang, Jie Hao, Zheng Chen, Saurabh Gupta, Xing Fan, and Chenlei Guo. 2021. Personalized search-based query rewrite system for conversational ai. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 179–188.

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)*.

Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context-and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 383–392.

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. In *2nd International Workshop on Data-Efficient Machine Learning (DeMaL)*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

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.

Pan Li, Zhen Qin, Xuanhui Wang, and Donald Metzler. 2019. Combining decision trees and neural networks for learning-to-rank in personal search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2032–2040.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. Setrank: Learning a permutation-invariant ranking model for information retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 499–508.

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 *AAAI*.

Zhen Qin, Zhongliang Li, Michael Bendersky, and Donald Metzler. 2020. Matching cross network for learning to rank in personal search. In *Proceedings of The Web Conference 2020*, pages 2835–2841.

Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2021. Are neural rankers still outperformed by gradient boosted decision trees? In *International Conference on Learning Representations*.

Hui Su, Xiaoyu Shen, Rongzhi Zhang, Fei Sun, Pengwei Hu, Cheng Niu, and Jie Zhou. 2019. Improving multi-turn dialogue modelling with utterance rewriter. *arXiv preprint arXiv:1906.07004*.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. In *ACL*.

Zhuoyi Wang, Saurabh Gupta, Jie Hao, Xing Fan, Dingcheng Li, Alexander Hanbo Li, and Chenlei Guo. 2021. Contextual rephrase detection for reducing friction in dialogue systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1899–1905, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270.

Xianchao Wu, Ander Martinez, and Momo Klyen. 2018. Dialog generation using multi-turn reasoning neural networks. In *NAACL*.

Siyang Yuan, Saurabh Gupta, Xing Fan, Derek Liu, Yang Liu, and Chenlei Guo. 2021. Graph enhanced query rewriting for spoken language understanding system. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7997–8001. IEEE.

A Appendix

A.1 Additional Experimental Setup

Evaluation Metrics We use *trigger rate* and *precision* as the precision metrics. In practice, the query rewriting system will not rewrite/trigger every query from the users to consider the cases such that the query itself may not be defected or the candidate list does not consider the correct rewrite. Thus, the ranking model will only trigger the query with certain confidence. In detail, the ranking model obtains a list of ranking scores corresponding to the list of candidates for each query. If the largest ranking score in the list of ranking scores exceeds a certain threshold, we trigger the query and use the top 1 candidate as the rewrite. Then the trigger label for this query is 1. For the query with largest ranking score below a certain threshold, the trigger label is 0. Mathematically speaking, given N queries $\{q_i\}$ and a list ranking scores $\{s_i^j\}_{j=1}^m$ [2] for each query q_i, the trigger rate is $\frac{\sum_{i=1}^N \mathbb{I}\{\max_j s_{i,j} \geq \tau\}}{N}$ where τ is the threshold. For the triggered query, if the top 1 candidate is a correct rewrite (label = 1), the prediction is correct. Thus, the precision is ratio of the number of queries that a model give correct prediction to the number of triggered queries. Mathematically speaking, let $y_{i,j} \in \{0, 1\}$ denote the label for candidate p_i^j and query q_i, $\forall i \in [N], j \in [m]$. The precision is defined as $\frac{\sum_{i=1}^N \mathbb{I}\{y_{i,\arg\max_j s_{i,j}}=1, \text{ and } \max_j s_{i,j} \geq \tau\}}{\sum_{i=1}^N \mathbb{I}\{\max_j s_{i,j} \geq \tau\}}$.

Datasets in Section 5.1 For training data, we use the weak-labeled data similar to Fan et al. (2021). For the retrieval phrase, from de-identified historical interactions of a large-scale conversational AI agent, we find two consecutive user utterances, where the first turn was defected and the second turn was successful by utilizing a defect detection model Gupta et al. (2021). For ranking, each query has 5 rewrite candidates retrieved from an index with FAISS search (Johnson et al., 2017). For each query, the 5 candidates are also labeled as *positive* (label = 1) and *negative* (label = 0), i.e., positive means the candidate is a correct rewrite and negative means the opposite. For this dataset, each query has at least one positive candidate among the 5 candidates, i.e., correct rewrite. We collect the training and test set from different time periods, i.e., 1-month data for training and subsequent 1-week data for testing. Moreover, after getting the raw test set, we have manually identified the true labeled cases and removed the noise. Number of queries of the dataset is provided in Table 6.

[2] In our experiment, we specify $m = 5$, i.e., 5 candidates for each query.

	Training	Validation	Test
No. queries	10.0x	1.0x	3.0x

Table 6: Summary of the preliminary ranking models datasets in Section 5.1. We report relative sizes with respect to the validation set.

Datasets in Section 5.2 In this set of experiment, we consider a more realistic dataset that includes all types of real-word queries. We refer to this dataset as Full-Scale Data. To be specific, different from the data set used in preliminary ranking models' experiments, here we consider 1) queries that have context information and 2) queries that do not have a positive candidate retrieved by the retrieval model. Each sample is comprised of a query, its dialogue context between the user and the conversational AI agent, and 5 candidates for rewrite. In practice, for some queries, there is no correct rewrite among the 5 candidates provided by the retrieval model. For those queries without correct rewrite in the candidate list, it is hard to qualify the performance of the ranking model due to the inherent data constraint. Thus, we specifically consider a subset of the original dataset named **Has-Rewrite**, i.e., the set of queries that has correct rewrites in the candidate list. For each query, the 5 candidates are also labeled as *positive* (label = 1) and *negative* (label = 0), i.e., positive means the candidate is a correct rewrite and negative means the opposite. We use the same way to extract and construct the training and test sets as we described before. Table 4 provide an example of this full-scale dataset with labels. Number of queries of the dataset is provided in Table 7.

A.2 Case Study

To further understand the advantage of context information for the context-aware ranking model, we provide examples of queries and the results of CANR and PWNR in Table 8. The major difference between CANR and PWNR is that CANR has utilized context information. In the first example, the user asked, "do you have reminders today you" which is defected. Then we can see that the CANR model can pick the correct rewrite "any reminders today". However, the PWNR model fails to pick

Full-Scale Data					
Training		Validation		Test	
All	Has-Rewrite	All	Has-Rewrite	All	Has-Rewrite
15.0x	10.0x	2.0x	1.0x	2.0x	1.0x

Table 7: Summary of the Full-Scale Data in Section 5.2. We report relative sizes with respect to the "Has-Rewrite" validation set.

Context	Retrieval candidates	Model predictions
USER: kitchen lights on AGENT: ok USER: do you have reminders today AGENT: George, here is your upcoming reminder. George, you have one reminder today. Time to take medications for our baby Tommy., everyday at 8:50 PM USER: do we have notification today AGENT: You don't have any notifications. USER: do you have reminders today you **(query)**	any reminders today do you have do i have any reminders can you reminders today no reminders today do you have reminders	CANR: any reminders today PWNR: do you have reminders
USER:who sings this AGENT: This is Fugazi . USER: play modern music AGENT:The station: Country Heat Radio. USER: play america's got talent music AGENT: Six Ways to Bomb on America's Got Talent by Melinda Hill , on Amazon Music. USER: play music from america con talent **(query)**	music from america's got talent play music by america play imagine from america's got talent play music from america play american's got talent music	CANR: music from america's got talent PWNR: play music from America

Table 8: Case study of comparing the context-aware neural ranking model CANR with PWNR. The correct rewrite (positive candidate) is underlined. The CANR can pick the correct rewrite since it incorporates the context information in the ranking.

up the correct rewrite. The important information "today" is missed by PWNR model. Similarly, in the second example, PWNR model has missed the information "got talent" in the rewrite. For both cases, we can see that that important information, i.e., "today" and "got talent", has appeared many times in the queries' context, which explains the good performance of the context-aware model.