PAIGE: Personalized Adaptive Interactions Graph Encoder for Query Rewriting in Dialogue Systems

Daniel Biś
Amazon Alexa AI
bisdb@amazon.com

Saurabh Gupta [1]
LinkedIn
saurabh3949@gmail.com

Jie Hao
Amazon Alexa AI
jieha@amazon.com

Xing Fan
Amazon Alexa AI
fanxing@amazon.com

Chenlei Guo
Amazon Alexa AI
guochenl@amazon.com

Abstract

Unexpected responses or repeated clarification questions from conversational agents detract from the users' experience with technology meant to streamline their daily tasks. To reduce these frictions, Query Rewriting (QR) techniques replace transcripts of faulty queries with alternatives that lead to responses that satisfy the users' needs. Despite their successes, existing QR approaches are limited in their ability to fix queries that require considering users' personal preferences. We improve QR by proposing **P**ersonalized **A**daptive **I**nteractions **G**raph **E**ncoder (PAIGE). PAIGE is the first QR architecture that jointly models user's affinities and query semantics end-to-end. The core idea is to represent previous user-agent interactions and world knowledge in a structured form — a heterogeneous graph — and apply message passing to propagate latent representations of users' affinities to refine utterance embeddings. Using these embeddings, PAIGE can potentially provide different rewrites given the same query for users with different preferences. Our model, trained without any human-annotated data, improves the rewrite retrieval precision of state-of-the-art baselines by 12.5–17.5% while having nearly ten times fewer parameters.

1 Introduction

Facilitating seamless human-computer interactions is a fundamental goal of conversational AI agents such as Alexa, Cortana, and Siri. However, some user interactions lead to *frictions*, where the AI agent delivers an unexpected response or repeatedly asks the user to clarify the query. Such frictions stem from system errors such as Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU). Some aspects of the frictions are highly personalized, depending on characteristics such as the user's demographics and interests. For example, when asking a conversational agent to "Put on Skyfall," one user may expect the system to play a song named "Skyfall" while another may wish to see a movie with the same title.

Query Rewriting (QR; Grbovic et al., 2015; Ponnusamy et al., 2020) aims to reduce frictions by replacing the transcripts of faulty queries with alternatives that lead to desired responses. Personalized QR systems were proposed in (Fan et al., 2021; Cho et al., 2021), which restricted rewrite candidates to the particular user's historical requests. Such systems, discussed in Section 6, typically trained a text encoder to measure the similarity between request and rewrite. While effective, they overlook correlations between requests within a user's dialogue history and the inter-dependencies spanning across other users' interactions. This information can help reformulate defective and ambiguous requests when augmented with external knowledge.

To address the aforementioned limitations, we introduce a QR architecture named **P**ersonalized **A**daptive **I**nteractions **G**raph **E**ncoder (PAIGE) that jointly models query semantics, world knowledge, and users' preferences in an end-to-end fashion. The core idea is to represent users' previous interactions in a heterogeneous graph that we can augment with external world knowledge (§3). The graph representation learning with Graph Neural Networks (GNNs) allows us to propagate the representations of users' historical interactions to refine the utterance embeddings in an end-to-end manner through joint training.

To construct the heterogeneous graph, we decompose the requests into smaller semantic units such as domains, intents, utterances, entities, and NLU-hypotheses (§3). We also create nodes representing the users and link every user node to nodes representing entities (e.g., songs, artists) appearing in the user's historical requests. Inspired by work in recommendation systems, we use cross-user connections to capture communicative intents among users (Goldberg et al., 1992; Wang et al., 2019b)

[1] Work done while at Amazon Alexa AI.

Proceedings of EMNLP 2022 Industry Track, pages 408–418
December 9–11, 2020.

and further ground the entity nodes in a knowledge graph (*e.g.*, Wang et al., 2019b, 2020), allowing PAIGE to learn from the emerging high-order connectivities.

We cast query rewriting as a link prediction problem between an utterance node and nodes corresponding to NLU-hypotheses, that abstract away syntactic variations. Once the link to NLU-hypothesis node is predicted, we can follow the graph's edges to select the most frequent non-defective utterance mapping to that NLU-hypothesis as our rewrite.

PAIGE is scalable in training, without the need to load the full graph in memory, and efficient at inference, without the need to re-process the entire graph. Our inductive node encoding scheme permits dynamically updating the graph to new knowledge and user interests without model re-training. We demonstrate the efficacy of our system with a detailed analysis of experiments on real-world conversation data (§5);

Our contributions are summarized as follows:

- We introduce PAIGE—a novel graph-based architecture for the task of personalized query rewriting in dialogue systems.
- We present a scalable and inductive method for joint learning of query semantics and structured user preferences in an end-to-end fashion.
- We show that modeling the high-order relations in the graph facilitates collaborative learning from customers' collective behaviors.
- PAIGE outperforms state-of-the-art baselines (*i.e.*, 43.8% P@1 increase) while having nearly $10\times$ fewer parameters.

2 Preliminaries

Spoken dialogue systems consist of many sequential components. When a user interacts with their device, the agent's ASR takes the audio signal as input and transcribes it into textual utterance (*query*). Next, the transcript enters the NLU module that interprets it so that the downstream modules can satisfy the user's request. An NLU component typically consists of domain and intent classification and entity linking, executed sequentially. As a preprocessing step for later modules in the dialogue system, the NLU module is instrumental to the system's overall quality. One of the challenges in the NLU module is handling ambiguity or errors cascading from the previous components. *Query Rewriting* (QR) component tackles this issue by replacing the ASR transcript with an alternative that leads to a satisfactory response for the user. Once the NLU pipeline receives a rewrite, regular data flow resumes.

2.1 Interactions Data Selection

As hand-annotating a large set of query-rewrite pairs is expensive, we use weakly-labeled data during training. Inspired by Fan et al. (2021) and Cho et al. (2021), we leverage users' feedback to collect the datasets. For example, if a user barged in or stopped the agent's response, the turn is defective. The details are available in Appendix A.

3 Graph Construction

The first step is building a heterogeneous graph from user-agent interactions expressed as text and semi-structured metadata. The graph will provide the computational architecture for the message passing algorithm.

Heterogeneous Graph (HG; Sun and Han (2013))**.** HG is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a node type mapping function $\tau : \mathcal{V} \rightarrow \mathcal{A}$ and a link type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$, where each node $v \in \mathcal{V}$ belongs to one particular node type $\tau(v) \in \mathcal{A}$, each link $\varepsilon \in \mathcal{E}$ belongs to a particular relation $\psi(\varepsilon) \in \mathcal{R}$, $|\mathcal{A}|+|\mathcal{R}| > 2$, and if two links belong to the same relation type, the two links have the same starting node types and ending node types.

3.1 Design Motivation

A simple way to build an interactions graph would be to link users with their utterances and the defective utterances with their rewrites. Unfortunately, such an approach produces a sparse graph due to the high degree of linguistic variance in the utterances and fails to capture users' entity and domain level preferences. However, GNNs require sufficient connectivity to be effective because their efficacy stems from feature propagation and smoothing across the graph's edges (Zhang et al., 2021).

NLU-Hypothesis. To abstract away syntactic variance in users' requests, we group queries with similar meaning by parsing them into structured representations called NLU-hypotheses using the agent's NLU module. Each hypothesis takes the form of "domain | intent | slot_type:slot_value." The domain is the general topic of a query, *e.g.*, "Weather." The intent reflects the action the user wants to take, *e.g.*, "PlayMusic." Finally, the slot types/values

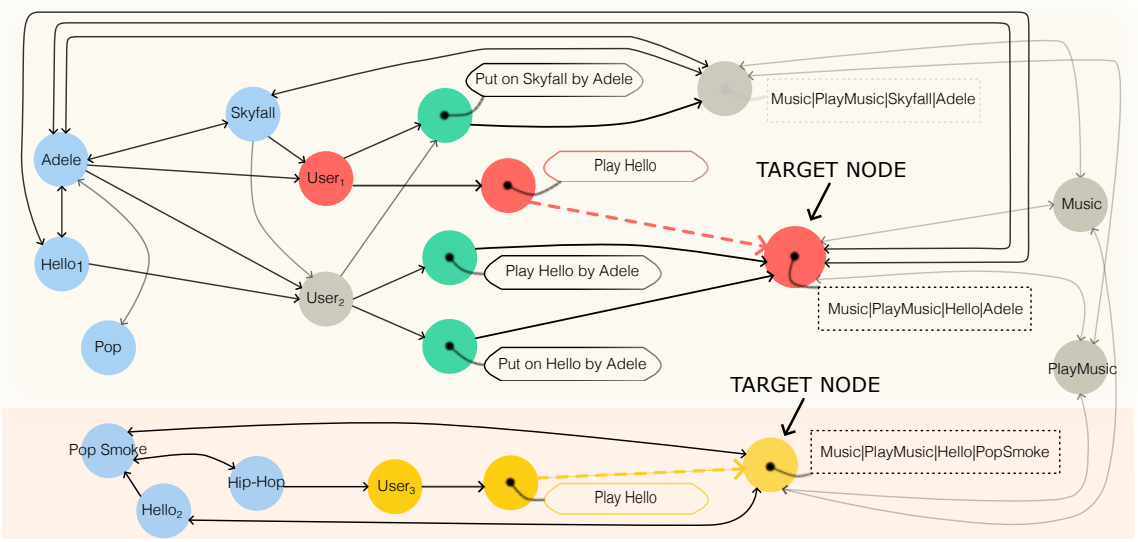

Figure 1: Customer interactions graph augmented with external knowledge. Knowledge enhances collaborative learning across users to enable reasoning-powered affinity/preference prediction. The goal is to map the two occurrences of an ambiguous utterance, *Play Hello*, to their respective interpretations (NLU-Hypotheses) for $user_1$ (red nodes) and $user_3$ (yellow nodes). To achieve this, PAIGE utilizes message passing over paths such as (Hello, Adele, $user_1$, *Play Hello*), and (Hello, Pop Smoke, $user_3$, *Play Hello*) for the two users. Nodes for non-defective queries are shared among users, *e.g.*, *Play Skyfall by Adele* (top, green).

are results of entity labeling from the NLU module. To illustrate, the queries *"Play Hello by Adele"* and *"Put on Hello by Adele"* map to the same hypothesis: "Music | PlayMusic | SongName:Hello | ArtistName:Adele." We use the hypotheses' fields as "semantic units" and assign them nodes to induce a dense graph with a rich set of relations.

3.2 Graph Schema

Every distinct hypothesis, $h \in \mathcal{H}$, is assigned a node in the graph. Moreover, we create a node for each unique domain, intent, and entity *(slot)*, and link them to the nodes representing the hypotheses in which they occur. Additionally, as illustrated in Figure. 1, edges in our graph connect users, $\mathcal{U}$, to their respective utterances, $\mathcal{T}$, and the utterances to their corresponding hypotheses, $\mathcal{H}$. The NLU-hypothesis nodes act as sub-graph pooling nodes and represent groups of equivalent queries and their side information, whereas the utterance nodes represent the individual queries. The utterance nodes do not need to be stored after training; instead, they can be created on the fly to keep the adjacency matrix up to date since PAIGE uses inductive encoders for nodes with textual input features (§4.1).

There are two types of utterance nodes in our graph: non-defective and defective. Including defective queries in the graph allows to explicitly model users' rephrase behaviors. We use historical query-rewrite pairs to replace the hypotheses of defective queries with the ones generated for their respective rewrites. In general, utterances with different NLU hypotheses map to different nodes, even if they have the same text, *e.g.*, the two "Play Hello" nodes in Figure 1. For *non-defective* queries, $c \in \mathcal{C}$, with identical text and NLU-hypothesis, we create a single utterance node to represent their text for all users, *e.g.*, a single "Play Skyfall by Adele" node in Figure 1 is shared by two users. For a *defective* utterance, $b \in \mathcal{B}$, we create a distinct *defect* node for each user for whom the utterance caused friction. Consequently, each defect node has a single incoming edge from the author's user node, and only the information relevant to the defective query's author directly affects the embedding of that query. At inference, we create a new defect node for an utterance that is not found in the user's dialogue history. Our task is to predict links between the new utterance nodes and the nodes associated with their NLU-hypotheses. Once the NLU-hypothesis node is predicted, we can simply follow the graph's edges to select the most frequent non-defective utterance that maps to that NLU-hypothesis as our rewrite.

Factual Knowledge. We align the entity nodes in our graph with nodes in a knowledge graph (KG). A KG is an instance of a heterogeneous graph that consists of real-world entities and their relation-

ships. A KG is organized into (v_i, r, v_j)-triples, where v_i, $v_j \in \mathcal{V}$ are the entities, and $r \in \mathcal{R}$ is the relation type, *e.g.*, *(Adele,* `AUTHOR OF`*, Hello)*.

Grounding the model in an explicit representation of knowledge facilitates rewrites that require understanding relationships that are not obvious from the user's dialogue history alone, *i.e.*, users' implicit preferences. We link the nodes corresponding to named entities found in each user's queries with the corresponding user nodes. As a result, the information from KG propagates through the user nodes to the utterance nodes. Crucially, as described above, entities are also connected to the NLU-hypotheses. Since GNN acts as neighborhood smoothing, our model favors the NLU-hypotheses with neighborhoods that contain entities relevant to the user who submits the query.

4 PAIGE Model

Computing node representations involves two steps: using specialized encoders to generate nodes' input features and applying message-passing layers to enable the features to interact and coalesce. In the message-passing step, we use relation-specific convolutional modules that aggregate feature vectors of the neighboring nodes. These modules learn to aggregate information from the node's immediate neighborhood, and stacking K such operations effectively convolves features across the K-th order neighborhood, *i.e.*, representations of nodes depend on all the nodes that are at most K edges away.

4.1 Input Features

PAIGE uses dedicated encoders for different types of nodes to produce input embeddings for the GNN. Our inductive feature encoding design permits updates to the graph's structure without expensive model retraining, *i.e.*, adding nodes for users, entities, and utterances. Thus, PAIGE can adapt to evolving user interests and world knowledge.

A major scalability challenge for end-to-end training of our model lies in encoding textual inputs. The reason is that the number of utterances needed to produce embeddings for a graph's nodes grows exponentially with the number of GNN layers. Therefore, we encode textual inputs, $t_i \in \mathcal{T}$, for the utterance nodes, $\tau(v_i) \in \mathcal{T}$, using a lightweight, two-layer Bidirectional Gated Recurrent Unit (BiGRU) network.

The domain and intent nodes are the only node types for which we use a fixed vocabulary; this is feasible because both sets change infrequently and amount to fewer than 10 K vectors. While previous works tend to use fixed-size embedding tables for users or entities (Wang et al., 2019b), such an approach prohibits dynamic updates to the graph's structure (*e.g.*, adding new users). Instead, we embed historical queries using a pre-trained RoBERTa-base model (Liu et al., 2019), and represent: *i) users* with the mean of embeddings of their previous queries; *ii) entities* with the mean of embeddings of queries in which they appear. The parameters of RoBERTa are fixed during training to prevent temporary trends in training data from leaking into initial entity representations.

Formally, a feature encoder enc_{τ_θ} embeds a node $v \in \mathcal{V}$ with type $\tau(v) \in \mathcal{A}$ as $\boldsymbol{x}_\tau = enc_{(\tau)_\theta}(v)$, where $\boldsymbol{x}_\tau \in \mathbb{R}^{d_\tau}$ is a dense feature vector. As the nodes come from different distributions, each feature encoder contains a fully connected feed-forward network that is applied to each node separately and identically to project vectors to a shared embedding space before the GNN layers.

$$\boldsymbol{h}_\tau^{(0)} = \phi\Big(\boldsymbol{x}_\tau \boldsymbol{W}_\tau^{(1)} + \boldsymbol{b}_\tau^{(1)}\Big)\boldsymbol{W}_\tau^{(2)} + \boldsymbol{b}_\tau^{(2)}, \quad (1)$$

where $\boldsymbol{h}_\tau^{(0)} \in \mathbb{R}^{d_{gnn}}$ is a node embedding, and $\boldsymbol{W}_\tau^{(1)} \in \mathbb{R}^{d_\tau \times d_{ffn}}$, $\boldsymbol{W}_\tau^{(2)} \in \mathbb{R}^{d_{ffn} \times d_{gnn}}$, $\boldsymbol{b}_\tau^{(1)} \in \mathbb{R}^{d_{fnn}}$, $\boldsymbol{b}_\tau^{(2)} \in \mathbb{R}^{d_{gnn}}$ are learnable parameters, and ϕ is a GELU activation (Hendrycks and Gimpel, 2016). We train feature encoders, except for RoBERTa, jointly with the graph neural network to enable each module to learn from other modalities.

4.2 Graph Encoder

In each layer, PAIGE propagates latent node feature information across edges of the graph while taking into account the type of an edge (Schlichtkrull et al., 2018). A single message-passing layer takes the following form

$$\bar{\boldsymbol{h}}_i^{(k+1)} = \phi\left(\sum_r \sum_{j \in N_r^i} \frac{\boldsymbol{W}_r \boldsymbol{h}_j^{(k)}}{c_{ij_r}} + \frac{\boldsymbol{h}_i^{(k)}}{c_{i_r}}\right), \quad (2)$$

where $\bar{\boldsymbol{h}}_i^{(k)} \in \mathbb{R}^{d_{gnn}}$ is the hidden state of node v_i in the k-th layer of the neural network, r is a relation type, $\boldsymbol{W}_r^{(k)} \in \mathbb{R}^{d_{gnn} \times d_{gnn}}$ is a relation-type specific parameter matrix, ϕ is a Leaky-ReLU activation, and c_{i_r} and c_{ij_r} are normalization constants.

To avoid over-smoothing, we apply residual con-

nections around each GNN layer,

$$\boldsymbol{h}_i^{(k+1)} = \alpha^{(k)}\mathrm{GNN}(\boldsymbol{h}_i^{(k)}) + (1-\alpha^{(k)})\boldsymbol{h}_i^{(k)}, \tag{3}$$

where $\alpha^{(k)}$ is a learnable scalar parameter.

Finally, we concatenate the representations of utterance nodes, $t \in \mathcal{T}$, from the BiGRU and GNN and pass the result through a feedforward network,

$$\boldsymbol{h}^{out}_{v_i \in \mathcal{T}} = \phi([\mathrm{BiGRU}(v_i) \parallel \mathrm{GNN}(v_i)]\,\boldsymbol{W} + \boldsymbol{b}) \tag{4}$$

where $[\cdot \parallel \cdot]$ is concatenation, $\boldsymbol{W} \in \mathbb{R}^{(2 \cdot d_{gnn}) \times d_{gnn}}$ and $\boldsymbol{b} \in \mathbb{R}^{d_{gnn}}$ are parameters, and ϕ is a Leaky-ReLU activation. Our experiments show that concatenating the input and output embeddings of the GNN improves the QR performance by up to 5.5%.

4.3 Graph Decoder

Once the encoder maps each node $v_i \in \mathcal{V}$ to an embedding, $\boldsymbol{h}_i$, the goal of the decoder is to use these embeddings to predict labeled links in the graph. In particular, the decoder scores a (v_i, r, v_j)-triplet using a function g to represent how likely it is that the hypothesis associated with v_j is the right interpretation of the utterance associated with v_i.

$$g(v_i, r, v_j) = \sum_m \left(\boldsymbol{h}_{i_m} \odot \boldsymbol{r} \odot \boldsymbol{h}_{j_m}\right), \tag{5}$$

where $\odot$ is element-wise multiplication, $r \in \mathbb{R}^{d_{gnn}}$ is a parameter vector, $\boldsymbol{h}_i, \boldsymbol{h}_j \in \mathbb{R}^{d_{gnn}}$ are embeddings of the source and target nodes, respectively.

4.4 Model Training

We train PAIGE on the link prediction task using binary cross-entropy loss with negative sampling. We sample N negative targets for each observed triple in the training set. By sharing the negative samples within each batch of size B, we obtain $N \times B$ negative targets for each positive triple.

The adjacency list and the feature matrix for the nodes reside in CPU memory due to their large memory footprint. We uniformly sample a fixed number of neighbors to convolve over in *eq.* 2 to control the memory consumption. The training procedure employs multiple CPU processes for neighborhood sampling, subgraph construction, feature extraction, and negative sampling, which then feed the constructed mini-batches to GPUs running model computations in parallel.

Data Split	Train	Dev	Test	Human Annotated
# of Examples	630K	89K	178K	1K

Table 1: Data summary.

Model	Precision@N (%)		
	P@1	P@5	P@10
RoBERTa	28.3	47.1	54.5
PAIGE-BoW	34.9	57.3	65.0
PAIGE-BiGRU	36.9	59.0	66.6
PAIGE (BiGRU ‖ GNN)	**40.8**	**64.5**	**71.8**

Table 2: PAIGE outperforms the RoBERTa-based baseline. The PAIGE-BoW and PAIGE-BiGRU remove the concatenation of the representations from the text and graph encoders; PAIGE-BoW uses Bag-of-Words encoder instead of BiGRU.

We compute RoBERTa embeddings for historical utterances offline and place them in Redis in-memory data store. The in-memory storage provides the CPU workers with fast access to the input features for a minibatch. This allows us to avoid repeated computation of utterance embeddings needed to produce the initial representations for the user and entity nodes.

4.5 Inference

Our graph design offers highly efficient inference as the representations of nodes added for incoming utterances do not affect other nodes in the graph. For an utterance node created at runtime, the only outgoing edge is the self-loop, and the only incoming edge is from the author's user node. Thus, we can cache the representations for the users' nodes from each GNN layer and compute the convolutions in eq. 2 only for the new utterance node. We cache the representations of the NLU-hypothesis nodes multiplied with the decoder's relation parameter vector, r in eq. 5, and use efficient Maximum Inner Product Search to select the rewrite.

5 Experiments

Here we evaluate PAIGE and empirically validate its performance. We begin with a quantitative assessment on general QR, followed by an evaluation on personalized use-cases.

5.1 Experimental Setup

Given a defective query, our task is to retrieve relevant rewrites from a large pool of candidates. The embeddings inferred by a model for a given user's

Model	User Index		Global Index		
	P@1	P@5	P@1	P@5	P@10
RoBERTa	78.0	98.2	28.9	47.0	54.27
PAIGE	**82.1**	**98.4**	**43.41**	**66.5**	**73.6**

Table 3: Rewrite precision (%) in a *mock* setting where all target rewrites are among the previous queries of the user who issued the query. Performance gains from our model increase when rewrite candidates are not limited to the user's previous queries (*User Index*) and a *Global Index* storing queries from all users is used.

	RoBERTa	**PAIGE**
Precision@1	77.9%	**79.4%**

Table 4: Rewrite precision on the human annotated dataset with rewrite candidates confined to individual users' past queries.

queries are evaluated on future rewrite actions of that user. Table 1 summarizes our datasets. We construct the datasets following the procedure detailed in Section 2.1 and Appendix A. The implementation details and hyperparameters are available in Appendix B. We follow previous works and evaluate models using Precision@N (P@N) metrics. The P@N measures if at least one rewrite among the first N retrieved candidates matches the target's utterance or NLU hypothesis. We implement the retriever from Fan et al. (2021) as our baseline but replace the Deep Structured Semantic Models (DSSM; Huang et al., 2013) with a pre-trained RoBERTa-base encoder to make it stronger. The baseline neural encoder takes a query's text as input and learns to minimize cosine distance between the embeddings of the query and the rewrite. The pre-trained model is fine-tuned on the dataset used to train PAIGE.

5.2 Results

Table 2 shows that PAIGE outperforms the baselines by 12.5–17.3%, indicating the efficacy of our personalized query embeddings. We observe the largest absolute gain of 17.3% for P@10, and the largest *relative* gain of 43.8% for P@1.

Query's semantics and graph's topology are complimentary. Ablation results in Table 2 show that feeding query embeddings from GNN to the decoder without first concatenating them with utterance representations from GRU results in a large drop in performance of up to ∼5.5%. Removing the concatenation step *and* replacing GRU with less expressive Bag-of-Words (BoW) encoder for textual input features decreases performance only by additional ∼2%. Finally, all graph-based models outperform the RoBERTa-based baseline.

Model	Precision@N (Relative change %)		
	P@1	P@5	P@10
RoBERTa	25.7	47.3	55.7
PAIGE	**28.5** (+10.9)	**54.5** (+15.3)	**63.0** (+13.1)

Table 5: Rewrite precision (%) on a dataset of query–rewrite pairs such that the rewrites do not appear in the user's dialogue history.

PAIGE improves personalized QR. To evaluate how well representations from PAIGE reflect users' proclivities, we evaluate models on a dataset of defective queries with rewrites that are among the user's previous requests. We consider two settings: *1)* limiting rewrite candidates to individual users' past queries (*User Index*), and *2)* using a global candidate index storing queries from all users (*Global Index*). The former setting is an easier task as the indexes for individual users typically contain ∼ 100 utterances while the latter uses a global index that contains ∼4.5M unique requests mapping to ∼2.2M hypotheses.

Table 3 shows that both models work well when the index is confined to the user's past queries. Notably, PAIGE offers nearly 4% higher P@1, opening promising avenues for future work on consolidating the retrieval and ranking steps into a single model. The performance gap increases dramatically when attempting rewrites from a global index storing queries from *all* users — *i.e.*, compared to RoBERTa, PAIGE improves P@1 by 14.4% and P@10 by 19.3% (Table 3, *Global Index*).

For a more comprehensive evaluation of PAIGE, we evaluate models on a test set consisting of query–rewrite pairs identified by human annotators (1K examples). As in the previous paragraph, the human annotated dataset consists of defective queries for which rewrites can be found among the individual users' historical requests. We confined rewrite candidates to the user's past queries (*User Index* setting). PAIGE achieves 79.4% P@1 on this test set compared to 77.9% from RoBERTa (Table 4).

PAIGE generalizes to unseen user preferences. To check if our model generalizes to unseen user preferences, we evaluate it on a set of query – rewrite pairs such that the rewrites do not appear in the user's dialogue history. We use examples

from entertainment domains that contain entities like songs and movies and tend to reflect users' affinities. Table 5 shows that our model offers up to 15.3% relative precision improvement over the baseline in this setting. This result is noteworthy since most traffic comes from entertainment domains.

6 Related Work

Non-personalized QR. Several prior studies have investigated the QR problem in a non-personalized context. Statistical QR models have been deployed in Alexa (Ponnusamy et al., 2020) and Google voice search (Sodhi et al., 2021). In their seminal work, Ponnusamy et al. (2020) apply an Absorbing Markov Chain (AMC) model as a collaborative filtering mechanism to mine reformulation patterns from sequences of user queries. At inference, an exact text match with a defect query in the index mined offline triggers a rewrite to the corresponding reformulation. Although statistical QR models are efficient at inference, they are transductive — limited to a fixed set of utterances — and do not generalize to unseen queries. Building on the work of Ponnusamy et al. (2020), Yuan et al. (2021) replace the Markov Chain with a GNN to capitalize on the distributed query representations, however, their method is still transductive. To facilitate inference on unseen queries, Chen et al. (2020) train a RoBERTa (Liu et al., 2019) encoder on a QR corpus. Other than the lack of personalization, the main limitation of these NR methods is that they treat each interaction independently, with side information encoded implicitly in the model's parameters. Recent studies show that performance of such methods tends to suffer when inputs contain rare words (Schick and Schütze, 2020; Biś et al., 2021) or spurious patterns (McCoy et al., 2019) such as common misconceptions (Podkorytov et al., 2021). PAIGE, on the other hand, uses the rich connectivities within the interactions graph and KG to refine the query representations.

Personalized QR. Fan et al. (2021) propose a Neural Retrieval NR-based personalized QR system. Through A/B testing on Alexa traffic, they demonstrate that the personalized approach improves user satisfaction relative to the non-personalized baseline. Fan et al. (2021) build a unique index for each user from the user's personal query log. They also build a global index storing historical queries from all customers. For each index type, dedicated neural encoders are trained to retrieve rewrite candidates, which are then ranked by an arbitration model. Cho et al. (2021) extend personalization to the ranker, providing it with user-specific features. As in the case of non-personalized NR, these models rely on user-agnostic query embeddings. In comparison, PAIGE selects rewrites using query representations that depend on users' prior experiences.

Graph Neural Networks. GNNs use input graph structures as computational architectures that aggregate neighborhood information to produce contextual representations for the nodes (Kipf and Welling, 2017; Schlichtkrull et al., 2018). In recommendation systems, GNNs operate on collaborative knowledge graphs that combine user-item interactions and structured knowledge (Wang et al., 2019b, 2020) to predict users' interests. These methods model the relations between interactions to learn from the customers' collective behaviors and alleviate issues caused by sparsity in interactions data (Wang et al., 2019b). While these studies tend to use bipartite graphs, PAIGE supports any graph structure. Other works use GNNs to model language and KGs together (Ghazvininejad et al., 2018; Talmor et al., 2019; Yang et al., 2020; Zhang et al., 2022). In comparison, PAIGE jointly models the language, knowledge, and user interactions.

7 Limitations

PAIGE enables rewrite retrieval from a global set of reformulation candidates but not all defects will be covered by the index. Considering this, a generative approach to the problem (Roshan-Ghias et al., 2020) offers an advantage but generative models pose quality control challenges in production systems, where issues like hallucinations (Lee et al., 2018) could have harmful effects.

8 Conclusion and Future Work

We put forward a graph-based framework for learning user affinities from their interactions with conversational AI agent. The proposed framework learns directly from user feedback and requires no human annotated data. Through extensive experiments on real-world conversations, we demonstrate that our proposed PAIGE improves the performance of QR systems and, as a result, reduces friction in users' interactions with the AI agent.

References

Daniel Biś, Maksim Podkorytov, and Xiuwen Liu. 2021. Too much in common: Shifting of embeddings in transformer language models and its implications. In *NAACL*.

Zheng Chen, Xing Fan, and Yuan Ling. 2020. Pre-training for query rewriting in a spoken language understanding system. In *ICASSP*.

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. *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*.

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)*.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

David Goldberg, David A. Nichols, Brian M. Oki, and Douglas B. Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35:61–70.

Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context-and content-aware embeddings for query rewriting in sponsored search. In *SIGIR*.

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)*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

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. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, page 2333–2338, New York, NY, USA. Association for Computing Machinery.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Katherine Lee, Orhan Firat, Ashish Agarwal, Clara Fannjiang, and David Sussillo. 2018. Hallucinations in neural machine translation. In *Interpretability and Robustness in Audio, Speech, and Language Workshop, (NeurIPS 2018)*.

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*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448.

Maksim Podkorytov, Daniel Biś, and Xiuwen Liu. 2021. How can the [mask] know? the sources and limitations of knowledge in bert. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Pragaash Ponnusamy, Alireza Roshan Ghias, Chenlei Guo, and Ruhi Sarikaya. 2020. Feedback-based self-learning in large-scale conversational ai agents. In *AAAI*.

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

Timo Schick and Hinrich Schütze. 2020. Bertram: Improved word embeddings have big impact on contextualized model performance. In *ACL*.

M. Schlichtkrull, Thomas Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. *ArXiv*, abs/1703.06103.

Sukhdeep Sodhi, Ellie Ka-In Chio, Ambarish Jash, Santiago Ontan'on, Ajit Apte, Ankit Kumar, Ayooluwakunmi Jeje, Dima Kuzmin, Harry Fung, Heng-Tze Cheng, Jonathan J. Effrat, Tarush Bali, Nitin Jindal, Pei Cao, Sarvjeet Singh, Senqiang Zhou, Tameen Khan, Amol Wankhede, Moustafa Alzantot, Allen Wu, and Tushar Chandra. 2021. Mondegreen: A post-processing solution to speech recognition error correction for voice search queries. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Yizhou Sun and Jiawei Han. 2013. Mining heterogeneous information networks: a structural analysis approach. *Acm Sigkdd Explorations Newsletter*, 14(2):20–28.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019a. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019b. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 950–958.

Ze Wang, Guangyan Lin, Huobin Tan, Qinghong Chen, and Xiyang Liu. 2020. *CKAN: Collaborative Knowledge-Aware Attentive Network for Recommender Systems*, page 219–228. Association for Computing Machinery, New York, NY, USA.

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.

Shiquan Yang, Rui Zhang, and Sarah Erfani. 2020. GraphDialog: Integrating graph knowledge into end-to-end task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1878–1888.

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*.

Wentao Zhang, Yuezihan Jiang, Yang Li, Zeang Sheng, Yu Shen, Xupeng Miao, Liang Wang, Zhi Yang, and Bin Cui. 2021. Rod: reception-aware online distillation for sparse graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2232–2242.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. GreaseLM: Graph REASoning enhanced language models. In *International Conference on Learning Representations*.

A Data Collection

We first find two consecutive user utterances in which the first turn was defective and the second was successful. To this end, we use defect detection models from Ling et al. (2020) and Gupta et al. (2021), and rephrase detectors from Wang et al. (2021). Moreover, we gather rewrites based on n-best hypotheses from ASR. In particular, if two consecutive user utterances are submitted less than t seconds apart and the first turn's ASR n-best $(n > 1)$ is the same as the second turn's ASR 1-best, we deem them rephrases. Finally, we apply rule-based filters to reduce noise in the data. We retain examples for which the edit distance between the two turns is less than d, and the time gap between the two turns is shorter than t seconds. We empirically set the $t = 45$, $d = 7$ to reduce noise and maintain the opportunity.

B Implementation Details

We implement PAIGE using Deep Graph Library (DGL) (Wang et al., 2019a). We set the GNN's dimension $(d_{gnn} = 256)$ and number of GNN layers $(L = 3)$. We use the batch size $(B = 512)$ and the number of negative samples per training example $(N = 32)$. With the in-batch sharing of corrupted triples, this results in $(N \times B = 16384)$ negatives per example. The parameters of the model are optimized by AdamW with weight decay $1e-3$. We use a warmup procedure that linearly increases the learning rate from $0.$ to $1e-3$ over the first 2000 training steps. Afterward, the learning rate decreases following the values of the cosine function. We use a dropout rate of 0.1 applied to each layer (Srivastava et al., 2014). The models are trained using eight GPUs (NVIDIA V100) with total memory of 256GB, which takes $\sim$20 hours on average.

C Additional Results

In Table 6 we report examples of faulty queries for which PAIGE provides correct rewrites but the baseline is unable to correct the defects.

Defective Query	RoBERTa Rewrite	PAIGE Rewrite
play pop junior clean	play pop culture radio clean	play pop clean
play downtown baby	play bad romance	play down down baby
what is the weather in	what is the weather today	what is the weather in utah

Table 6: Examples of defective queries where PAIGE provides correct rewrites, but the baseline fails to correct the defects.

D Relationships in the Interactions Graph

We provide additional details on the node and relation types in PAIGE interactions graphs. Table 7 summarizes node counts and their respective features and feature encoders. Table 8 describes the main relations captured by the graph's edges.

# of Nodes	Count†	Features	Encoder	End-to-End
Utterance	4.4M	Text	BiGRU	√
Defect-Utterance	635K		+ Adapter	√
NLU-Hypothesis	2.2M	Frequency & Defect-rate	MLP + Adapter	√ √
User	400K	Mean embedding of user's queries	RoBERTa	×
Entity	650K	Mean embedding of queries *w/*entity	+ Adapter	√
Domain	38	Embedding Matrix	Embeddings	√
Intent	6722		+ Adapter	√

Table 7: Our node encoding strategy allows graph updates without model re-training, *e.g.*, for new knowledge or users. †The node counts do not include nodes added *on-the-fly* during evaluation.

Interaction	Linking Entities	Link Type	Description
Affinity	Entity-User	Directed	A user interacts with an entity
Authorship	User-Utterance	Directed	A user submitted an utterance
Realization	Utterance-Hypothesis	Directed	An utterance expresses the hypothesis
Rewrites	DefectUtt.-Hypothesis	Directed	A defect-uttr. maps to non-defective hypothesis
Slot-Value-Rel	Entity-Hypothesis	Bi-Directed	An entity appears in hypothesis
Domain of	Domain-Hypothesis	Bi-Directed	A domain appears in hypothesis
Intent of	Intent-Hypothesis	Bi-Directed	A intent appears in hypothesis
Attribute	entity-entity	Bi-Directed	A relationship between entities

Table 8: Relationships in PAIGE's Interactions Graph.