GRAPH ENHANCED QUERY REWRITING FOR SPOKEN LANGUAGE UNDERSTANDING SYSTEM

Siyang Yuan [*]

Duke University,
Electrical and Computer Engineering

Saurabh Gupta, Xing Fan, Derek Liu,
Yang Liu, Chenlei Guo

Amazon.com Inc., USA

ABSTRACT

Query rewriting (QR) is an increasingly important component in voice assistant systems to reduce customer friction caused by errors in a spoken language understanding pipeline. These errors originate from various sources such as Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) modules. In this work, we construct a user interaction graph from their queries using data mined from a Markov Chain Model [1], and introduce a self-supervised pre-training process for learning query embeddings by leveraging the recent developments in Graph Representation Learning (GRL). We then fine-tune these embeddings with weak supervised data for the query rewriting task, and observe improvement over the neural retrieval baseline system, demonstrating the effectiveness of the proposed method.

Index Terms— Query rewrite, neural retrieval, contextual embedding, pre-training, user friction, graph representation learning

1. INTRODUCTION

For advanced voice assistants like Alexa, Google Home and Siri, Spoken Language Understanding (SLU) systems serve as a core component as they are responsible for interpreting semantic meaning of the voice input. The SLU systems usually consist of two parts: an ASR component to convert voice signals to text and an NLU component to extract information ready to be used in downstream tasks. In a real product, each of the two components could introduce errors, e.g. ASR component could misinterpret the audio signal due to background noise and NLU component system could misconstrue the semantic meaning due to ambiguity. Such errors could accumulate and might result in users rephrasing requests or abandoning their tasks, which we refer to as user *friction*. Frequent occurrence of such *friction* cases results in unsatisfactory user experience. Thus, a mechanism to reduce this *friction* is central to SLU component in voice assistant.

Motivated by query rewriting models in web search[2, 3, 4, 5, 6], several works have been proposed in SLU system for friction reduction, including [7, 8, 1, 9, 10]. In our paper, we define *query rewriting* (QR) task as a way to replace the original ASR interpretation of an input utterance spoken by a user (*query*) with the most appropriate *rewrite* within candidates' library that improves the overall success rate of SLU system. For example, in Figure 1, given the initial defective query "play walk by cardi b" and customer's own rephrases "play warp by cardi b" and "play wap by cardi b", the QR model is expected to learn from customer feedback and provide a rewrite of "play wap by cardi b" as Cardi B has a song named "*Wap*", and neither "*Warp*" nor "*Walk*".

[*]This work was completed while the first author was an intern at Amazon.

<u>**Utterance**</u> (turn 1): play walk by cardi b
Domain: Music | **Intent**: PlayMusic | **SongName**: walk | **ArtistName:** cardi b
<u>**Utterance**</u> (turn 2): play warp by cardi b
Domain: Music | **Intent**: PlayMusic | **SongName**: warp | **ArtistName**: cardi b
<u>**Utterance**</u> (turn 3): play wap by cardi b
Domain: Music | **Intent**: PlayMusic | **SongName**: wap | **ArtistName**: cardi b

Fig. 1. An interaction session of utterances and their NLU hypotheses.

In order to improve the QR model's generalization ability, Chen et al. [10] proposed to pre-train the query embedder with large scale existing historical sessions data and users' *implicit feedback*. Specifically, a neural retrieval model based on Siamese neural network is adopted and the session data from customer is truncated as bi-gram segment for pre-training process. However, this model can successfully learn to re-write only if the following request is a correct rephrase of the previous one. As shown in Figure 1, the model will learn to rewrite "*warp*" to "*wap*", but would fail to map "*walk*" to "*wap*".

To mitigate this, we extend the bi-gram model to construct an interaction graph of users' queries in order to leverage longer contextual information into the pre-training process. We adopt the Markov chain models proposed in [1] and generate condensed representation to represent customer interaction with SLU system. The pre-training is conducted on sampled paths from the interaction graph using GCN [11]. The introduction of interaction graph enables us to store the historical interactions in graph form, where the topological structure could capture the transitioning in user's intention at multi-hop level. The method is flexible enough and can be extended to incorporate different node and edge features. Compared with bi-gram models, the GCN collectively aggregates information from graph structure through message passing from neighboring nodes that are multiple hops away, which enables the node representations to capture dependencies across user interactions. Once the query embeddings are learned with this pre-training method, we evaluate their effectiveness on the downstream query rewriting task. Finally, we provide some discussions of the proposed method.

2. METHODOLOGY

In this section, we first describe the query rewriting task and the interactions graph. Afterwards, we present our proposed framework to pre-train the query embeddings with the available interactions graph. Then we describe the fine-tuning procedure on QR using the pre-trained embeddings.

2.1. Downstream Query Rewriting Task

In this paper, following [10], the QR problem is formed as an information retrieval (IR) task. Given a query and millions of indexed rewrite candidates, the model is required to select the most relevant candidate as the query's rewrite. The retrieval system is designed such that the neural encoder learns to encode latent syntactic and semantic information from the given query specifically for the QR task. In this way, we can compute the fixed size vector representation for the large number of rewrite candidates offline, which is ideal for large-scale voice assistant.

2.2. Interactions Graph

Graphs are a ubiquitous data structure, usually represented with $G =< V, E >$, where V is the set of vertices or nodes and E is the set of edges. Here, we use a graph to represent users' historic interactions with the voice assistant and subsequently learn the query representations that encode structural information about the graph. Since users' utterances have a high degree of semantic and structural variance [1], we map the utterances to the **NLU hypotheses** from the NLU system, which consists of results from domain and intent classifiers as well as the named entity recognition (NER) model. As described later, we compute the embeddings of these NLU hypotheses in pre-training, which we refer to as *query embeddings*. As shown in Figure-2, the nodes of the interactions graph are NLU hypotheses of users' queries and edges capture transition probabilities across them. These transition probabilities are conditional probabilities computed using the counts of NLU hypotheses pairs across all users, corresponding to pairs of successive utterances within 45 seconds of each other. The figure shows how the users rephrase or repeat their queries until the voice assistant gets it right. For example, "*Play fearless*" sometimes plays a random *fearless* video, so the user rephrases and adds "*by Taylor Swift*".

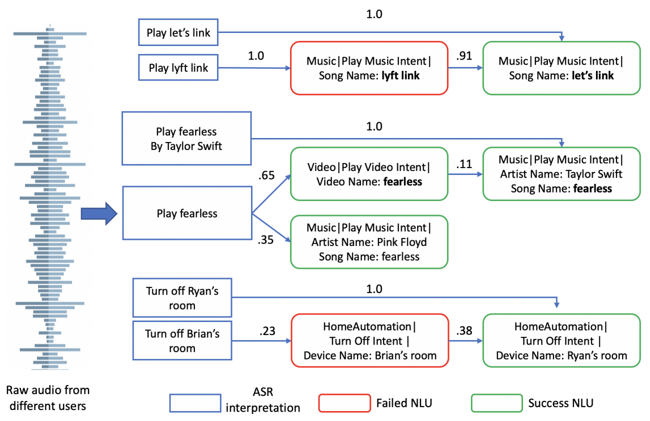
Play let's link
1.0
Play lyft link
1.0
Music|Play Music Intent| Song Name: lyft link
.91
Music|Play Music Intent| Song Name: let's link
Play fearless By Taylor Swift
1.0
Play fearless
.65
Video|Play Video Intent| Video Name: fearless
.11
Music|Play Music Intent| Artist Name: Taylor Swift Song Name: fearless
.35
Music|Play Music Intent| Artist Name: Pink Floyd Song Name: fearless
Turn off Ryan's room
1.0
Turn off Brian's room
.23
HomeAutomation| Turn Off Intent | Device Name: Brian's room
.38
HomeAutomation| Turn Off Intent | Device Name: Ryan's room
Raw audio from different users
ASR interpretation
Failed NLU
Success NLU

Fig. 2. An interactions sub-graph with nodes as NLU hypotheses, and edges with the transition probabilities

2.3. Self supervised Pre-training Query Embedding with interactions graphs

QR model requires a considerable amount of training data to train a model with good performance and generalization. However, annotated QR data is expensive to get. Thus, pre-training query embedding with existing interactions data could infuse structural information that captures user transitions into query representations.

Graph representation learning (GRL) aims to encode nodes as low-dimensional vectors that summarize their graph position and the structure of their local graph neighborhood. These embeddings can be viewed as projections in a latent space where geometric relations correspond to interactions (e.g., edges) in the original graph [12]. We investigate pre-training with three GRL methods: Deepwalk[13], ConvKB[14] and GraphSAGE[15]. We also report results of Word2Vec [16] as a baseline. For different models, we use different strategies to include the transition probabilities in consideration during training. For *Word2Vec*, we sample paths from the interactions graph based on the transition probabilities. Each path is treated as a sentence, with individual queries being treated as words. So, in this case, the transition probabilities get naturally encoded from the sampling process.

Deepwalk[13] is a two stage model which first traverses the graph with random walk, and then uses the SkipGram Word2Vec[16] model to learn embeddings. Instead of sampling random paths with uniform distribution, we use the transition probabilities for path sampling. *ConvKB*[14] is designed for knowledge graph completion(KGC), in which the edges have its own attributes. The vectorized embedding of head and tail nodes are fused with a convolutional layer, thus the scoring function is calculated in a non-linear way. In training, the loss function is calculated as

$$L = y \cdot f(\mathbf{z}_h, \mathbf{z}_t), \tag{1}$$

in which $y = r_{ht}$ for the linked pairs, $y = -r_{ht}$ for the sampled negative pairs. r_{ht} represents the transitioning probability from head node h to tail node t.

GCN [17] based models are ideal for leveraging the topological structure of graph, in which node embedding is represented with an aggregation of neighborhood nodes' embedding, especially in comparison with ConvKB which only focus on pairs of linked nodes. However, all the methods described so far are inherently transductive, i.e., they can generate representations only for the nodes present during training. So, we also explore *GraphSAGE*[15] , which is an inductive framework that leverages node feature information and can generate representations for unseen data. This framework also works better on large scale graphs, as it first samples a neighborhood within the graph, and is then trained in a GCN like manner. For GraphSAGE, we use the graph completion objective function introduced in [15] for training. The loss function is defined as

$$L = -r_{ht} \cdot \log\left(\sigma(\mathbf{z}_h^\top \mathbf{z}_t)\right) - r_{ht} \cdot \mathbb{E}_{t_n \sim P_n(h)} \log\left(\sigma(-\mathbf{z}_h^\top \mathbf{z}_{t_n})\right), \quad (2)$$

in which $\mathbf{z}_h$ represents head node mebedding, $\mathbf{z}_t$ represents tail node embedding, t_n is negative tail node, sampled from $P_n(h)$ (a negative sampling distribution), r_{ht} is the transitioning probability from head node h to tail node t.

With the above models, NLU hypotheses' embeddings are trained with interactions graph, thus containing information of transitioning probability of intents of users. The detailed experimental setups are introduced in Section 3.2.

2.4. Fine-tuning on Query Rewriting Task

The core component of our query rewriting model is an encoder architecture referred to as *embedder*. Given a raw input query $\mathbf{u}$, we first extract the NLU hypotheses with certain NLU system, and then compute the corresponding pretrained query embedding $\mathbf{z}$ as described in Section 2.3. The features of raw utterances $\mathbf{x}$, which is another component of query, are computed with pre-trained BERT[18].

The two vectors are fused with an attention mechanism. They are first fed into a single-layer MLP to project them into the same feature space, and then a weighted sum is computed, further followed by a three-layer MLP to compute the final embedding:

$$\text{Emb}(u) = \text{linear}_{\text{all}}(a_z \cdot \text{linear}_z(\mathbf{z}) + a_x \cdot \text{linear}_x(\mathbf{x})). \quad (3)$$

The attention weights are calculated using:

$$a_z = \frac{\exp(\text{linear}_{att_z}(\mathbf{z}))}{\exp(\text{linear}_{att_z}(\mathbf{z})) + \exp(\text{linear}_{att_x}(\mathbf{x}))}, \quad (4)$$

and a_x is computed in a similar manner.

In order to measure similarity between query $\mathbf{u}$ and rewrite $\mathbf{u}'$, we use cosine distance:

$$\Delta(\mathbf{u}, \mathbf{u}') = \cos(\text{Emb}_1(\mathbf{u}), \text{Emb}_2(\mathbf{u}')). \quad (5)$$

We use margin loss as training objective:

$$L = \max(0, \alpha - \Delta(\mathbf{u}, \mathbf{u}') + \Delta(\mathbf{u}, \mathbf{u}'_{-})), \quad (6)$$

in which $\mathbf{u}'_{-}$ is the sampled unpaired data of query $\mathbf{u}$. This loss function would encourage higher similarity score between paired data, and lower similarity score between unpaired data.

Another important component of the retrieval system is the k-Nearest Neighbor (kNN) index. All rewrite candidates' representations are computed offline and added to the KNN index implemented using FAISS[19]. At retrieval time, FAISS is used to retrieve the top-k relevant rewrites from the embedded space, using the cosine similarity given the raw input query.

3. EXPERIMENTS

3.1. Data

We construct two separate data sets. The one used to pre-train the hypotheses embedding is referred to as the "**pre-training set**", and the other one used to perform the downstream QR task is called the "**QR set**".

For the **pre-training set**, we extract historical sessions across all available users in a given time range. We only include the dialog sessions that *ended* successfully. Note that within the same session, there might be some unsuccessful turns. In the end, we collect over 17 million paths with 650K unique NLU hypotheses nodes.

For the **QR set**, the data is generated using an existing rephrase detection model pipeline. Within a session, we select query pairs: (u, u') that have high confidence score from the rephrase detection model. In each such pair, the second query u' is a corrected form of the first unsuccessful query u. This process gives a fine-tuning dataset (QR set) of 7.8 million rewrite pairs. The **QR test-set** is sampled from friction traffic (i.e. likely an error) to specifically measure error recovery and is particularly challenging.

3.2. Experimental Setup

Pre-training setup To directly evaluate the quality of pre-trained query embeddings, we try to retrieve the neighbours; given the head node, retrieve the tail node with highest transitioning probability among all potential tail nodes. Specifically, two metrics are reported to evaluate the performance: Precision@1 (p@1) and normalized discounted cumulative gain (NDCG). The p@1 measures if the retrieved candidate is the ground truth tail node, and NDCG is a measurement of overall ranking quality. We explored performance of several GRL methods, including DeepWalk, ConvKB, GraphSAGE, and Word2Vec. For *GraphSAGE*, initial embedding for each node is required. Thus, we use two types of initial features: the one trained with ConvKB(referred to as *GraphSAGE (ConvKB)*) and the one that uses encoded BERT representations of NLU hypotheses (referred to as *GraphSAGE (BERT)*).

QR experiment setup For downstream task of query rewriting, we compare the retrieved rewrite candidate's NLU hypotheses with the actual NLU hypotheses of the true rewrite

u' from the pair (u, u') in the test set. For each query, we retrieve the top 20 rewrites. We report the metric of Precision@N (p@n) using the retrieved rewrites' NLU hypotheses. Here, the p@n measures if at least one rewrite in the n candidates has a NLU hypotheses that matched the ground truth NLU.

Due to some graph pruning strategies used in [1], the overlap between pre-training data and QR data is limited. Thus, we provide results on the overlapping data to validate our idea, as well as testing results on full QR dataset. Overlapping dataset refers to the instances in the QR dataset, whose NLU hypotheses for both query and rewrite exist in the pre-training graph database. Under this setup, there are 2.9 million QR data instances in training, 491k data in testing.

For each setup, we report two sections of results, with and without fine-tuning on the QR dataset. Our baseline, referred to as ***Text2Text***, uses only the raw utterance features from BERT. It does not use the pre-trained embeddings of NLU hypotheses.

We use batch size of 2048 for all QR experiments, the index size for retrieval task is 4.5 million. Adam optimizer is used for training and the embedding space for *embedder* is set as 300.

3.3. Experimental Results

As discussed in Section 3.2, we first report the results of pre-training, followed by results of fine-tuning on both *overlapping* and full QR dataset.

Pre-training performance: Table 1 shows the retrieval performance on pre-training dataset. In comparison with word embedding based *Word2Vec* model, GRL models perform better. Among the GRL models, ConvKB model achieves the best performance. In terms of using only the node embeddings for retrieval task with dot product scoring function, GraphSAGE based model performs the best. Hence, we choose node embeddings trained with ConvKB and GraphSAGE in downstream QR tasks.

Table 1. Retrieval results on pre-training dataset. The first three rows are results using models' original scoring function (trained convolutional layer for ConvKB). The last three rows are results using dot product between nodes as scoring function. Precision and NDCG scores are absolute differences w.r.t the baseline.

Method	P@1	NDCG
Word2Vec	0% (range of 20%)	0 (range of 0.7)
Deepwalk	24.8%	0.106
ConvKB	71.5%	0.245
GraphSAGE (ConvKB)	36.9%	0.183
GraphSAGE (BERT)	32.5%	0.167
ConvKB	14.3%	0.083

Fine-tuning performance on QR overlapping set: Since the overlap between pre-training data and QR data is limited (only 30% of fine-tuning QR data has nodes existing in pre-training data), we first validate our approach on the overlapping data, as shown in Table 2. The proposed framework could achieve large performance gains in comparison with baseline model: *Text2Text*. This result confirms the idea that pre-training the model helps in learning the patterns in user queries that are transferable to downstream tasks.

Table 2. QR performance on the overlapping dataset. The first three rows show the results without fine-tuning. The other rows show the results with fine-tuning on QR dataset. Precision scores are absolute differences w.r.t the baseline.

Method	P@1	P@5	P@10	P@20
ConvKB	4.2	3.7	3.5	3.0
GraphSAGE (ConvKB)	0	0	0	0 (range of 10)
GraphSAGE (BERT)	0.3	0	0.1	0.2
Text2Text	7.6	28.4	34.3	37.6
ConvKB	**13.3**	**32.3**	**35.9**	38.1
GraphSAGE (ConvKB)	11.5	30.8	**35.9**	**38.4**
GraphSAGE (BERT)	11.2	30.4	35.3	37.7

Fine-tuning performance on full QR test set: Given that the idea is validated on the overlapping data, the results on whole dataset is shown in Table 3. Although for the full test dataset, the performance gain is not as significant as overlapping dataset, the proposed framework could still improve the precision on testing set. Furthermore, using a larger pre-training dataset for increasing the overlap with QR test set seems to be a promising direction for future work.

Table 3. Query Rewriting results on whole dataset. The first three rows show the results without fine-tuning, other rows show the results with fine-tuning on QR dataset. Precision scores are absolute differences w.r.t the baseline.

Method	P@1	P@5	P@10	P@20
ConvKB	2.62	2.50	2.55	2.33
GraphSABE (ConvKB)	0.00	0.00	0.00	0.00 (range of 10)
GraphSAGE (BERT)	0.15	0.00	0.04	0.15
Text2Text	11.15	29.06	34.85	**38.56**
ConvKB	**13.42**	**30.15**	35.34	38.23
GraphSAGE (ConvKB)	12.37	29.53	34.69	37.92
GraphSAGE (BERT)	12.76	30.04	**35.31**	**38.56**

4. CONCLUSION AND FUTURE WORK

This work proposes a neural retrieval based query rewriting approach to reduce user friction in a spoken language understanding system. The proposed QR model leverages graph representation learning of queries, pre-trained with a large amount of historical dialog sessions across all users. In comparison with the QR model without pre-trained embeddings, the proposed model shows distinct advantage in performance.

In future, this line of work could be extended to more downstream tasks like recommendations, where the voice assistant can proactively anticipate users' intentions and provide better experiences. Also the framework could be improved by more sophisticated graph design to incorporate attributes like success rate of NLU hypotheses, contextual features and demographics of users.

5. REFERENCES

[1] Pragaash Ponnusamy, Alireza Roshan Ghias, Chenlei Guo, and Ruhi Sarikaya, "Feedback-based self-learning in large-scale conversational ai agents," in *AAAI*, 2020, vol. 34, pp. 13180–13187.

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

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

[4] Stefan Riezler and Yi Liu, "Query rewriting using monolingual statistical machine translation," *Computational Linguistics*, vol. 36, no. 3, pp. 569–582, 2010.

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

[6] Jinxi Guo, Tara N Sainath, and Ron J Weiss, "A spelling correction model for end-to-end speech recognition," in *ICASSP*. IEEE, 2019, pp. 5651–5655.

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

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

[9] Dmitriy Serdyuk, Yongqiang Wang, Christian Fuegen, Anuj Kumar, Baiyang Liu, and Yoshua Bengio, "Towards end-to-end spoken language understanding," in *ICASSP 2018*. IEEE, 2018, pp. 5754–5758.

[10] Zheng Chen, Xing Fan, and Yuan Ling, "Pre-training for query rewriting in a spoken language understanding system," in *ICASSP*. IEEE, 2020, pp. 7969–7973.

[11] Thomas N Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[12] William L. Hamilton, Rex Ying, and Jure Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, 2017.

[13] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.

[14] Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al., "A novel embedding model for knowledge base completion based on convolutional neural network," in *NAACL*, 2018, pp. 327–333.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1024–1034.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, "Graph attention networks," in *ICLR*, 2018.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019, pp. 4171–4186.

[19] Jeff Johnson, Matthijs Douze, and Hervé Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, 2019.