RecQR: Using Recommendation Systems for Query Reformulation to correct unseen errors in spoken dialog systems

MANIK BHANDARI, Amazon Alexa AI, U.S.A
MINGXIAN WANG, Amazon Alexa AI, U.S.A
OLEG POLIANNIKOV, Amazon Alexa AI, U.S.A
KANNA SHIMIZU, Amazon Alexa AI, U.S.A

As spoken dialog systems like Siri, Alexa and Google Assistant become widespread, it becomes apparent that relying solely on global, one-size-fits-all models of Automatic Speech Recognition (ASR), Natural Language Understanding (NLU) and Entity Resolution (ER), is inadequate for delivering a friction-less customer experience. To address this issue, Query Reformulation (QR) has emerged as a crucial technique for personalizing these systems and reducing customer friction. However, existing QR models, trained on personal rephrases in history face a critical drawback - they are unable to reformulate unseen queries to unseen targets. To alleviate this, we present RecQR, a novel system based on collaborative filters, designed to reformulate unseen defective requests to target requests that a customer may never have requested for in the past. RecQR anticipates a customer's future requests and rewrites them using state of the art, large-scale, collaborative filtering and query reformulation models. Based on experiments we find that it reduces errors by nearly 40% (relative) on the reformulated utterances.

CCS Concepts: • **Information systems** → **Recommender systems**.

ACM Reference Format:
Manik Bhandari, Mingxian Wang, Oleg Poliannikov, and Kanna Shimizu. 2023. RecQR: Using Recommendation Systems for Query Reformulation to correct unseen errors in spoken dialog systems. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3604915.3610235

1 INTRODUCTION

Millions of customers interact with voice assistants to perform a variety of tasks ranging from turning on the lights to playing their favorite song. However, many such requests are not handled appropriately, leading to customer frustration. A bulk of these errors are one of ASR (Automatic Speech Recognition), NLU (Natural Language Understanding) or ER (Entity Resolution) errors. It is extremely challenging for global statistical ASR, NLU or ER models to work well for scenarios that are statistically rare such as a person with a unique accent. Moreover, customer requests can be inherently ambiguous like "Play Baby Shark" which can mean the video for some customer but the song for another, or "Play Courier" vs "Play Korea" which could be homophones for some customers.

One of the key, recent approaches for error correction via query reformulation is Cho et al. [6] which performs a search over an index of past, successful customer requests. For instance, if a customer requested to "Play Old Time Road" successfully in the past, it will store the utterance for the customer in the customer's index. Now for an incoming request, it finds the closest utterance from the customer's index to rewrite the customer request. In this example, because

Manuscript submitted to ACM

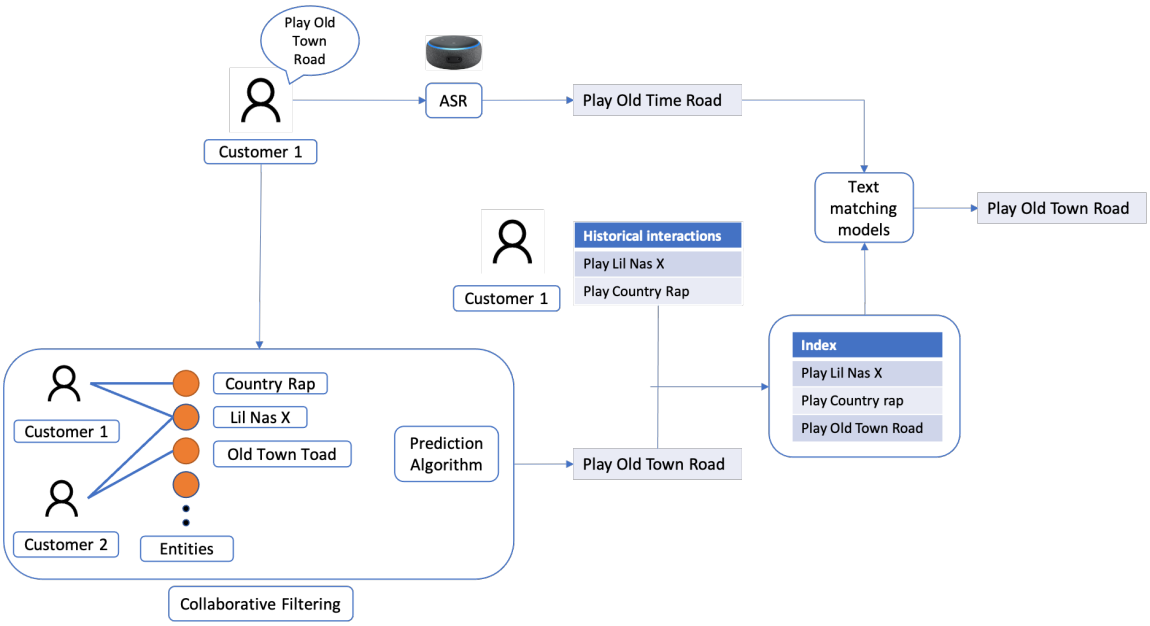


Fig. 1. RecQR overview.

"Play Old Time road" sounds similar to "Play Old Town Road", and the customer had requested successfully for "Play Old Town Road" in the past, the defective utterance is rewritten to "Play Old Town Road". However, this approach has a critical drawback - it solely relies on past customer interactions. Thus, if a customer requests for an entity for the first time and there's an error, Cho et al. [6] cannot correct it.

RecQR aims to alleviate this reliance on only using past interactions by anticipating future customer requests and adding them to the customer's personalized index. We do this by modeling the affinity of a customer with every entity (like the song "Old Town Road") - even if the customer has never requested for an entity in the past. We define the affinity between a customer and an entity, as the likelihood that the customer will request for that entity in the future. This affinity modeling is done via collaborative filtering [1] where we start with a set of historically high affinity entities - the entities that customers have requested for in the past repeatedly, with success and build a graph-based, large scale recommender system to find the novel entities that a customer is likely to request in the future.

To summarize, we make the following key contributions:

(1) We provide the first system of query reformulation that is capable of rewriting an unseen defective request to a target query which the customer may never have requested in the past.
(2) Our approach ties together the field of recommender systems and query reformulation to improve spoken dialog systems. This allows us to leverage the recent advancements made in the respective fields to improve spoken dialog systems.
(3) Through experiments and A/B tests, we demonstrate the efficacy of RecQR by showing that it scales to a graph with billions of edges and significantly reduces customer friction.

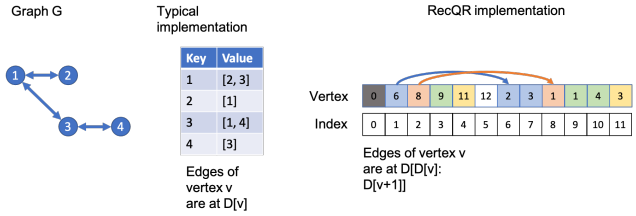
2 RELATED WORK

Query reformulation (QR) is a fast developing field with applications in question answering [4], machine translation [16], search advertising [14], ASR error reduction [5, 6, 15] among others. Most of the prior work focuses on correcting errors for previously seen queries. Ponnusamy et al. [15] creates a Markov chain based collaborative filtering method that mines the past interactions of users to find patterns where users reformulate their query successfully. This approach however, cannot reformulate unseen queries. To address this drawback, Fan et al. [8] proposes a search based QR system. The system introduces two indices, consisting of possible reformulations obtained from a customer's history, operating at both a global level (global index) and a personal level (personal index). During runtime, a neural network based scorer retrieves the top-k candidates for an input query, both from the global index and a customer's personal index. Then, a tree based ranker ranks the retrieved candidates and an arbitrator picks the target reformulation for a query. This allows for unseen queries to be reformulated to (previously seen) targets. Cho et al. [6] further extend this system to include a ranking layer for candidates retrieved from the personalized index. This personalized search based QR system however, relies solely on past customer interactions to create the customer's personal index, preventing it from rewriting a query to a target query that the customer has never requested for in the past. RecQR addresses this drawback and provides a method for rewriting unseen queries to targets that a customer may never have requested for in the past.

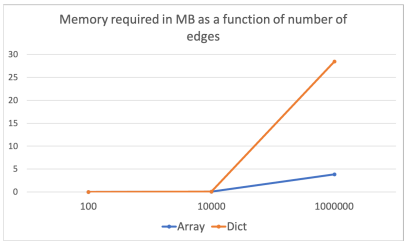
Collaborative filtering (CF), a widely recognized technique in recommender systems [12, 17, 21], has attracted considerable interest in literature [19, 20]. Traditional approaches [12, 17, 21] commonly rely on factorizing the user-item interaction matrix into a product of two lower dimensional matrices. This approach, however, does not scale easily to the massive scale of modern voice assistants. To address this drawback, several random walks based approaches for CF have been proposed [2, 3, 7, 9, 10, 18]. RecQR is a flexible personalized error correction system that can use any collaborative filtering system as long as it can scale to interaction graphs with billions of edges. In this work, we base our collaborative filtering method on Pixie [7] which is a state-of-the-art graph-based recommender system known for its real-time capability. While we retain the core idea of Pixie, we streamline the implementation to suit the specific requirements of query correction within a large-scale spoken dialogue system. For instance, Pixie leverages the textual content of the items for creating topic models used for pruning noisy nodes and edges in the graph. This is not possible for RecQR as our items do not have any content information.

3 RECQR OVERVIEW

We deploy RecQR for personalized error correction by leveraging Cho et al. [6], which consists of (1) text matching models and (2) a personalized cached index of utterances for each customer, to reformulate queries. During runtime, the text matching models search the personalized index to find the closest utterance to the input utterance. This closest utterance acts as the reformulation (a.k.a. rewrite) and is passed downstream. Figure 1 shows the data flow of a misrecognized utterance. RecQR's prediction algorithm consumes an interaction graph G consisting of customers and entities and applies collaborative filtering to predict future customer requests. In this example, it predicts that the customer might request "Play Old Town Road" in the future because other customers who request for "Lil Nas X" also request for "Old Town Road". These predicted future requests are stored along with historical customer requests to form a customer's personalized index. When an utterance is misrecognised as "Play Old Time Road", because RecQR has



(a) Multiple ways to represent G.



(b) Memory comparison of dict vs array implementation. Memory grows exponentially with the number of edges.

Fig. 2. Benefit of representing the interaction graph as a contiguous array.

predicted "Play Old Town Road" for this customer, text matching models are able retrieve and trigger a reformulation of a previously unseen query to a target that the customer had never requested in the past.

4 RECQR GRAPH CONSTRUCTION

We construct a heterogeneous interaction graph $G = \{V, E\}$ which consists of customers and entities as the vertices (V), and the interactions between them as edges (E). When constructing the graph, we add an undirected edge between a customer and an entity if the entity is a historically high affinity entity for the customer. Here, historical affinity is defined using a combination of frequency and recency from the historical interactions between a customer and an entity. The equation below defines the historical affinity of an entity on any given day. Equation 1 implies that affinity halves every 7 days.

$$\text{Affinity}(c, e, t) = \sum_{d=t-h}^{t} \left[\{N_{\text{success}}(c, e, d) - N_{\text{defect}}(c, e, d)\} \, 2^{\frac{-(t-d)}{7}} \right] \tag{1}$$

Here, c is a customer, e is an entity, t is the target date when we wish to calculate affinity and h is the history size i.e. the number of days of interactions to consider. N_{success} and N_{defect} are the number of successful and defective requests respectively made by customer c for entity e on day d. To estimate if a request for an entity is a success or a defect, we leverage error detection models like Gupta et al. [11], Ling et al. [13]. These error detection models use large-scale language models along with behavioral signals like whether a customer barged in to stop the voice assistant etc. to predict if there was an error when a user asked a query. Using Equation 1, we define historically high affinity entities for a customer as $\{e \in \text{Entities} \mid \text{Affinity}(c, e, t) > \theta\}$. Here Entities is the set of all entities and θ is a minimum threshold above which we consider an entity to be a historically high affinity entity.

5 EXPERIMENTS

5.1 Graph Representation

We represent G as an adjacency list but in the form of a contiguous array, instead of a dictionary. This is a highly scalable representation and allows fast graph traversal and efficient storage. See Figure 2a for an explanation of how we represent G as a contiguous array. This representation is critical because it allows the complete graph to fit in the main memory of most machines, avoiding distributed graph storage and traversal which is computationally inefficient. See Figure 2b to compare the memory requirement of a typical dict implementation against the efficient array representation.

Because of the efficient implementation, we are able to train RecQR and anticipate future customer requests on a daily cadence, allowing us to adapt to quickly changing customer preferences.

5.2 Datasets

We consider de-identified and anonymized user interactions from a commercial voice assistant over a time period of 30 days for training and 14 days for testing. We only keep customers who requested for at least 5 unique historically high affinity entities in the training set so that we have enough historical data to make meaningful predictions for a customer. For each customer, we only keep the top-100 entities to prevent dilution of information during the random walks. Our dataset consists of millions of customers (represented as an by an anonymized ID) interacting with millions of entities (represented by the name of the entity like "lil nas x"). We construct the train and test sets based on time - we keep interactions occurring for 30 days as the training set and for 14 days as the test set. We tune the hyperparameters on a completely different train and validation set to prevent any overfitting. For training, we end up with a massive graph of billions of edges but because of the efficient representation, it occupies only a small amount of memory which allows us to keep a full copy of the graph on each distributed node of a spark cluster [1].

6 RESULTS

In this experiment we calculate the recall of RecQR. Recall@N measures the fraction of future requests that we correctly anticipated, given an index of size N. As mentioned in Section 3, we are restricted by the personalized index size limit and evaluate RecQR for different index sizes in Table 1. For confidentiality concerns, we report relative improvements over a strong baseline of Cho et al. [6].

Model	**Recall@50**	**Recall@100**
RecQR	20.13%	32.11%

Table 1. Relative improvements in Recall@N of RecQR with varying index size N as compared against Cho et al. [6]

We also conducted an A/B test where we measured the performance of RecQR and found that it is able to reduce the error rate of affected traffic [2] by nearly 40% relative from 26.39% (control) to 16.81% (treatment) as measured by a defect detection model such as Gupta et al. [11], Ling et al. [13].

7 PRESENTER'S BIO

Manik Bhandari [3] is an Applied Scientist II at Amazon's Alexa AI team, based out of Boston, USA. At Alexa AI, Manik's research has focused on building large-scale recommender systems. Prior to this, Manik was a Master's student at Carnegie Mellon University [4] where he worked on the problem of evaluating automatically generated text. He was also a Research Assistant at Indian Institute of Science [5] where he worked on building as well as utilizing large-scale Knowledge Graphs for various Natural Language applications.

[1] https://spark.apache.org/
[2] Volume of affected traffic is ommitted due to confidentiality
[3] Manik's Publications: https://scholar.google.com/citations?user=_6pvM64AAAAJhl=en
[4] https://www.cmu.edu
[5] https://iisc.ac.in

REFERENCES

[1] [n. d.]. Collaborative Filtering. https://en.wikipedia.org/wiki/Collaborative_filtering. Accessed: 2022-05-31.

[2] Lars Backstrom and Jure Leskovec. 2011. Supervised Random Walks: Predicting and Recommending Links in Social Networks. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining* (Hong Kong, China) *(WSDM '11)*. Association for Computing Machinery, New York, NY, USA, 635–644. https://doi.org/10.1145/1935826.1935914

[3] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. 2010. Fast Incremental and Personalized PageRank. *Proc. VLDB Endow.* 4, 3 (dec 2010), 173–184. https://doi.org/10.14778/1929861.1929864

[4] Daniele Bonadiman, Anjishnu Kumar, and Arpit Mittal. 2019. Large Scale Question Paraphrase Retrieval with Smoothed Deep Metric Learning. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*. Association for Computational Linguistics, Hong Kong, China, 68–75. https://doi.org/10.18653/v1/D19-5509

[5] Zheng Chen, Xing Fan, Yuan Ling, Lambert Mathias, and Chenlei Guo. 2020. Pre-Training for Query Rewriting in A Spoken Language Understanding System. https://doi.org/10.48550/ARXIV.2002.05607

[6] 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*. Association for Computational Linguistics, Online, 179–188. https://doi.org/10.18653/v1/2021.nlp4convai-1.17

[7] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1775–1784. https://doi.org/10.1145/3178876.3186183

[8] 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 (De-MaL)*.

[9] Ashish Goel, Pankaj Gupta, John Sirois, Dong Wang, Aneesh Sharma, and Siva Gurumurthy. 2015. The Who-To-Follow System at Twitter: Strategy, Algorithms, and Revenue Impact. *Interfaces* 45, 1 (feb 2015), 98–107.

[10] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Zadeh. 2013. WTF: The Who to Follow Service at Twitter. In *Proceedings of the 22nd International Conference on World Wide Web* (Rio de Janeiro, Brazil) *(WWW '13)*. Association for Computing Machinery, New York, NY, USA, 505–514. https://doi.org/10.1145/2488388.2488433

[11] Saurabh Gupta, Xing Fan, Derek Liu, Benjamin Yao, Yuan Ling, Kun Zhou, Tuan-Hung Pham, and Chenlei (Edward) Guo. 2021. RoBERTaIQ: An efficient framework for automatic interaction quality estimation of dialogue systems. In *KDD 2021 Workshop on Data-Efficient Machine Learning*. https://www.amazon.science/publications/robertaiq-an-efficient-framework-for-automatic-interaction-quality-estimation-of-dialogue-systems

[12] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. 1997. GroupLens: Applying Collaborative Filtering to Usenet News. *Commun. ACM* 40, 3 (mar 1997), 77–87. https://doi.org/10.1145/245108.245126

[13] Yuan Ling, Benjamin Yao, Guneet Kohli, Hung Pham, and Chenlei (Edward) Guo. 2020. IQ-Net: A DNN model for estimating interaction-level dialogue quality with conversational agents. In *KDD Converse 2020*. https://www.amazon.science/publications/iq-net-a-dnn-model-for-estimating-interaction-level-dialogue-quality-with-conversational-agents

[14] Akash Kumar Mohankumar, Nikit Begwani, and Amit Singh. 2021. Diversity Driven Query Rewriting in Search Advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery I&; Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 3423–3431. https://doi.org/10.1145/3447548.3467202

[15] Pragaash Ponnusamy, Clint Solomon Mathialagan, Gustavo Aguilar, Chengyuan Ma, and Chenlei (Edward) Guo. 2022. Self-aware feedback-based self-learning in large-scale conversational AI. In *NAACL 2022*. https://www.amazon.science/publications/self-aware-feedback-based-self-learning-in-large-scale-conversational-ai

[16] Stefan Riezler and Yi Liu. 2010. Query Rewriting Using Monolingual Statistical Machine Translation. *Computational Linguistics* 36, 3 (Sept. 2010), 569–582. https://doi.org/10.1162/coli_a_00010

[17] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (Hong Kong, Hong Kong) *(WWW '01)*. Association for Computing Machinery, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071

[18] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast Random Walk with Restart and Its Applications. In *Proceedings of the Sixth International Conference on Data Mining (ICDM '06)*. IEEE Computer Society, USA, 613–622. https://doi.org/10.1109/ICDM.2006.70

[19] Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z. Sheng, Mehmet A. Orgun, Longbing Cao, Francesco Ricci, and Philip S. Yu. 2021. Graph Learning based Recommender Systems: A Review. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 4644–4652. https://doi.org/10.24963/ijcai.2021/630 Survey Track.

[20] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1, Article 5 (feb 2019), 38 pages. https://doi.org/10.1145/3285029

[21] Yong Zhuang, Wei-Sheng Chin, Yu-Chin Juan, and Chih-Jen Lin. 2013. A Fast Parallel SGD for Matrix Factorization in Shared Memory Systems. In *Proceedings of the 7th ACM Conference on Recommender Systems* (Hong Kong, China) *(RecSys '13)*. Association for Computing Machinery, New York,

NY, USA, 249–256. https://doi.org/10.1145/2507157.2507164