Accepted at AAAI 2024 Workshop on Synergy of Reinforcement Learning and Large Language Models

**Reinforcement Learning for Optimizing RAG for Domain Chatbots**

**Mandar Kulkarni, Praveen Tangarajan, Kyung Kim, Anusua Trivedi**

Flipkart Data Science

Seattle, Washington, USA

|  |  |  |
| --- | --- | --- |
| arXiv:2401.06800v1 [cs.CL] 10 Jan 2024 | **Abstract** | have become good at contextual question-answering tasks, |
| With the advent of Large Language Models (LLM), conversa- | i.e., given a relevant text as a context, LLMs can generate |
| answers to questions using that information. Retrieval Aug- |
| tional assistants have become prevalent for domain use cases. |
| mented Generation (RAG) is one of the key techniques used |
| LLMs acquire the ability to contextual question answering |
| to build chatbots for answering questions on domain data. |
| through extensive training, and Retrieval Augmented Genera- |
| tion (RAG) further enables the bot to answer domain-specific | RAG consists of two components: a retrieval model and |
| questions. This paper describes a RAG-based approach for | an answer generation model based on LLM. The retrieval |
| building a chatbot that answers user’s queries using Fre- | model fetches context relevant to the user’s query. The query |
| quently Asked Questions (FAQ) data. We train an in-house | and the retrieved context are then inputted to the LLM with |
| retrieval embedding model using infoNCE loss, and exper- | the appropriate prompt to generate the answer. Though in- |
| imental results demonstrate that the in-house model works |
| tuitive to understand, RAG-based chatbots have a few chal- |
| significantly better than the well-known general-purpose pub- |
| lenges from a practical standpoint. |
| lic embedding model, both in terms of retrieval accuracy and |
| Out-of-Domain (OOD) query detection. As an LLM, we use | 1. If a retrieval model fails to fetch relevant context, the gen- |
| an open API-based paid ChatGPT (gpt-35-turbo-16k-0613) | erated answer would be incorrect and uninformative. The |
| model. We noticed that a previously retrieved-context could | retrieval is more challenging for non-English queries, |
| be used to generate an answer for specific patterns/sequences |
| e.g., code-mix Hinglish. |
| of queries (e.g., follow-up queries). Hence, there is a scope |
| 2. For paid API-based LLMs (e.g., ChatGPT), the cost per |
| to optimize the number of LLM tokens and cost. Assuming a |
| call is calculated based on the number of input and out- |
| fixed retrieval model and an LLM, we optimize the number |
| of LLM tokens using Reinforcement Learning (RL). Specifi- | put tokens. A large number of tokens passed in a context |
| cally, we propose a policy-based model external to the RAG, | leads to a higher cost per API call. With a high volume |
| which interacts with the RAG pipeline through policy actions | of user queries, the cost can become significant. |
| and updates the policy to optimize the cost. The policy model |
| 3. To enable multi-turn conversations with RAG, a conver- |
| can perform two actions: to fetch FAQ context or skip re- |
| sation history needs to be maintained and passed to the |
| trieval. We use the open API-based GPT-4 as the evaluation |
| LLM with every query. It is known that a larger input to- |
| model, which rates the quality of the bot answer. Using an |
| ken size leads to a drop in accuracy or hallucinations as |
| appropriate reward shaping, GPT-4 ratings are converted to |
| a numeric reward. We then train a policy model using pol- | LLMs have an additional task of choosing relevant infor- |
| icy gradient on multiple training chat sessions. As a pol- | mation from a large context (Liu et al. 2023a). |
| icy model, we experimented with a public gpt-2 model and |
| In this paper, we first describe a RAG-based approach for |
| an in-house BERT model. With the proposed RL-based op- |
| building a chatbot that answers user’s queries using Fre- |
| timization combined with similarity threshold, we are able |
| quently Asked Questions (FAQ) data. We have a domain |
| to achieve significant (~31%) cost savings while getting a |
| slightly improved accuracy. Though we demonstrate results | FAQ dataset consisting of 72 FAQs regarding the credit |
| for the FAQ chatbot, the proposed RL approach is generic | card application process. The FAQ dataset is prepared to an- |
| and can be experimented with any existing RAG pipeline. | swer user queries regarding general card information pre- |
| **Introduction** | and post-application queries. We train an in-house retrieval |
| (embedding) model using info Noise Contrastive Estima- |
| tion (infoNCE) loss (van den Oord, Li, and Vinyals 2019) |
| With the advent of Large Language Models (LLM), we ob- |
| with the English and Hinglish paraphrase queries created |
| serve an increased use of conversational assistants even for |
| using ChatGPT. The embedding model is trained to maxi- |
| the domain use cases. Trained on a large web-scale text cor- |
| mize query-question and query-QnA similarity. Experimen- |
| pus with approaches such as instruction tuning and Rein- |
| tal results show that the in-house model performs signifi- |
| forcement Learning with Human Feedback (RLHF), LLMs |
| cantly better than a public pre-trained embedding model re- |
|  | Copyright © 2024, Association for the Advancement of Artificial | garding retrieval accuracy and Out-of-Domain (OOD) query |
| Intelligence (www.aaai.org). All rights reserved. | detection. OOD queries are questions unrelated to the do- |

main data, e.g., how is the weather today? For every user query, we retrieve top-k FAQs (question + answer) as con-text and input them to the LLM to generate the answer. We maintain the previous two queries, answers, and FAQ con-text as history to enable the multi-turn conversation. We use an open paid API-based ChatGPT (gpt-35-turbo-16k-0613) for all our experiments as an LLM. GPT-4 is used to rate the bot answer quality (as Good/Bad) for the end-to-end RAG pipeline.

We propose an RL-based approach to optimize the num-ber of tokens passed to the LLM. We noticed that for cer-tain patterns/sequences of queries, we can get a good answer from the bot even without fetching the FAQ context. Ex-amples of such scenarios can be: 1. for a follow-up query; FAQ context need not be retrieved if it has already been fetched for the previous query; 2. for the sequence of queries referring to the same FAQ, a context can be fetched only once at the start; 3. for OOD queries, the LLM prompt it-self can guide the bot to generate the answer. Using this in-sight, we propose a policy gradient-based approach to opti-mize the number of LLM tokens and, hence, the cost. The input to the policy model is the State, which comprises previous queries, previous policy actions, and the current query. The policy model can take two actions: [FETCH] or [NO\_FETCH]. A usual RAG pipeline would be executed when a policy network takes a [FETCH] action. When a policy network chooses a [NO\_FETCH] action, FAQ con-text is not retrieved. A query and context (empty in the case of [NO\_FETCH] action) are inputted to the LLM. We use GPT-4 as the reward model and convert the quality rating (Good/Bad) to the numeric reward using appropriate reward shaping. If the LLM generates a good answer (as rated by GPT-4) even without fetching the context, we give it a high positive reward to promote such actions in the future. If not fetching a FAQ context leads to a wrong answer, we pro-vide a negative reward (e.g., this can happen if the policy model chooses a [NO\_FETCH] action for a domain query without any previous relevant context). For the training chat sessions, for each State, a policy model samples an action according to the current probability distribution over the ac-tions. We then generate (State, Action, Reward) trajectories by sampling multiple times from the current policy. A pol-icy model is then updated using a policy gradient with cu-mulative reward. As a policy model, we experimented with in-house BERT and public gpt-2 models. Figure 1 shows the architecture of the proposed policy-based approach. Exper-imental results demonstrate that the policy model provides token saving by fetching the FAQ context only when it is re-quired. When combined with a simple similarity threshold-based optimization, we are able to achieve token savings of ~31% on the test chat session with 91 queries while achiev-ing slightly improved accuracy (evaluated through manual labeling) than the usual RAG pipeline. This underlines the effectiveness of the proposed approach.

**Related works**   
With recent advancements in Generative AI and LLMs, the Retrieval-Augmented Generation (RAG) (Lewis et al. 2021) approach has emerged as the preferred strategy for contex-

2

**Training in-house embedding model**

To use RAG for FAQ chatbots, we first need to retrieve the top k most relevant FAQs to answer the given query. To fetch top-k FAQs, an embedding model is used to encode queries and FAQs. We concatenate a question and an answer within a FAQ and use this to get a vector for that FAQ. A cosine sim-ilarity measure is then used to rank the FAQs based on their relevance. We experimented with popular general-purpose public pre-trained embedding models and models trained on in-domain data. We trained an in-house embedding model with infoNCE loss (van den Oord, Li, and Vinyals 2019) us-ing a well-known embedding model (e5-base-v2) weights as initialization. InfoNCE has proved an effective loss for con-trastive learning, specifically under self-supervised setting (Goyal et al. 2021). We finetune a publicly available embed-ding model *e5-base-v2* (Wang et al. 2022) using in-domain data with infoNCE loss.

We create a dataset for training the embedding using ChatGPT and small manual tagging. We generate multiple English queries per FAQ as question paraphrases. Due to the large non-English-speaking population in India, we observe many Hinglish queries on the platform. We also generate a few Hinglish paraphrase queries per FAQ to support code-mix Hinglish queries. Next, we add a small set of manually tagged corpus where we add queries that can be answered based on the FAQ answers’ content. We use ~3.5k queries for training, ~1k queries for validation, and 1014 queries for testing.

We use the following loss function to train the model.

|  |  |  |
| --- | --- | --- |
|  | exp(sim(*zi, zj*)*/τ*) | (1) |
| *li,j* = | *k*=11[*k̸*=*i*] exp(sim(*zi, zk*)*/τ*) |  |

Here, *zi* and *zj* indicate a positive pair (i.e., question para-phrases for the same FAQ), *B* indicates batch size, and *τ*indicates temperature. For all our experiments, we set *N* to 8 and *τ* to 0.1 and use cosine similarity for sim.

As seen from Eq. 1, infoNCE loss is specifically suit-able for training with query-FAQ mapped data because it only needs positive pairs and treats remaining samples as in-batch negatives. We finetune the embedding model with two objectives: maximize query-QnA similarity and query-question similarity.

Table 1 shows the top-1 and top-3 accuracy comparison results for English and Hinglish queries with a general-purpose public model (e5-base-v2), the models finetuned with triplet loss and infoNCE loss. The ranking results are computed based on query-QnA cosine similarity. It can be seen that the in-house model trained with infoNCE works significantly better than the public pre-trained model.

Next, we compare the detection performance of pub-lic and in-house models on in-domain vs out-of-domain (OOD) samples. We created a dataset of 30 in-domain and OOD queries each. In-domain queries contain English and Hinglish queries, while OOD queries include greeting, ac-knowledgment, and general non-domain-related questions. We calculated mean scores for positive and negative queries with the public and finetuned model. Table 2 shows the

3

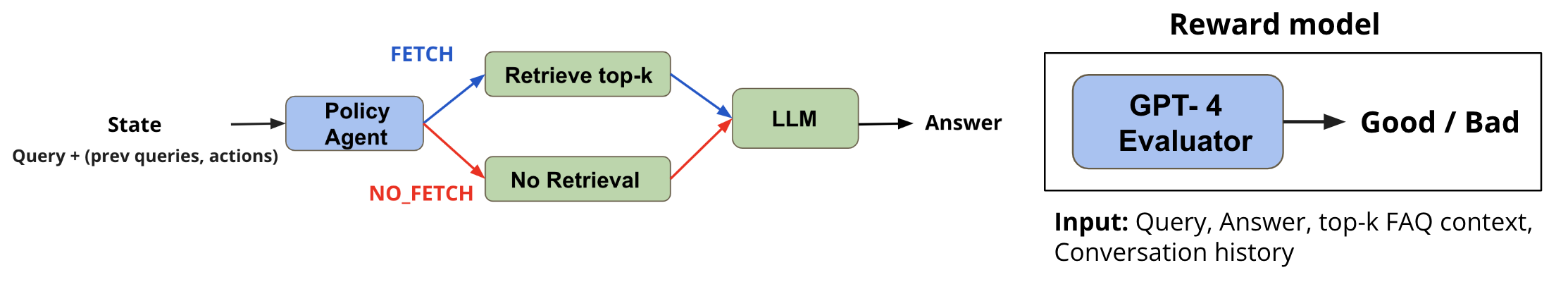


Figure 1: Proposed policy agent based architecture for optimizing RAG for domain chatbots.

with the current query, current FAQ context, answer gener-ated from the bot, and conversation history (which includes previous query, answers, and FAQ context). We do not pro-vide GPT-4 with the ideal or ground truth response. Instead, we prompt GPT-4 to evaluate the quality of the generated an-swer based on the information available in the current FAQ context and conversation history. GPT-4 is instructed to eval-uate the output based on whether it exactly matches the in-formation provided in the FAQ context. Any additions or de-viations from the available information are termed as factual failures and are to be rated bad. GPT-4 is asked to provide a final rating as good or bad and the reason behind the rating. For the OOD queries, the bot is supposed to provide a "do not know" answer, and GPT-4 is instructed to rate such an-swers as good. Table 3 and 4 show query, bot response, and GPT-4 evaluation examples.

|  |  |
| --- | --- |
| **query** | is there annual fee for card |
| **bot response** | Yes, there is an annual fee of INR 500 for the credit card. However, this fee will be waived off if your annual spends exceed INR 3.5 Lakhs. |
| **gpt4 eval** | Good |
| **gpt4 reason** | The answer is directly address-ing the query and providing the relevant information from the FAQ context. The answer matches with the information in the FAQ context, stating that there is an annual fee of INR 500 for the the credit card, which will be waived off if the annual spends exceed INR 3.5 Lakhs. |

Table 3: GPT-4 evaluation for an in-domain query

We evaluated the bot output on a chat session’s sequence of 25 queries. Queries consisted of English and Hinglish domain queries, OOD queries (e.g., hows the weather), greeting/acknowledgment queries, etc. The GPT-4 evalua-tion provided 100% accuracy for these answers. A manual verification of evaluations was done to re-confirm the re-sults. The evaluation indicated that the FAQ bot is able to respond accurately to user queries.

4

FAQ context only for the first query, and all the subse-quent queries can utilize this context. Hence, fetching FAQ context on every query is unnecessary under such a scenario.

• **OOD queries**:

LLM is prompted to answer OOD queries with a "do not know" answer. Hence, even if we fetch FAQ context for OOD queries, LLM is instructed to ignore it. Hence, fetching a context would not be needed for OOD queries as well.

As can be seen from the list above, there are many cases for which fetching FAQ context would not be needed to get the same answer as with the usual RAG pipeline and, there-fore, there is a scope to optimize the tokens passed to the LLM to reduce the cost without trading off the accuracy. We propose a Reinforcement Learning (RL) based ap-proach to optimize the number of tokens passed to LLM. Figure 1 shows the proposed approach for policy-based agent for optimizing RAG. Specifically, we train a policy network to maximize rewards from a GPT-4 evaluator. The policy network resides outside the RAG pipeline and tries to optimize it by tuning the policy. We assume that we do not have access to gradients of the retrieval model or LLM and treat these components as fixed.

Input to the policy network is the State, which comprises previous queries, the corresponding actions (encoded as to-kens), and the current query. Since our RAG pipeline uses the last two conversations as history, for the policy model as well, we maintain the previous two queries as context. The policy network can take two actions for the current query: [FETCH] and [NO\_FETCH]. As the name suggests, when a policy network chooses a [FETCH] action, a usual RAG pipeline would be executed. When a policy network chooses a [NO\_FETCH] action, FAQ context is not retrieved, and the current query is directly inputted to the LLM. The LLM then generates the answer. The current query, answer, and conversation history is inputted to a GPT-4, which rates the bot answer as good or bad.

Intuitively, if a policy model chooses a wrong action, in that case, GPT-4 should provide a rating as Bad, e.g., if a pol-icy model chooses a [NO\_FETCH] action for the in-domain query without any earlier context. We observe that it is in-deed the case, and Table 5 shows an example of such a case. The GPT-4 evaluation rating is converted to the numeric reward (*r*) value as depicted in Table 6. An intuition behind this reward function is that when a policy model executes a [NO\_FETCH] action and GPT-4 rates the bot response as Good, it indicates that for the input query not fetching a context still leads to a good output, possibly because one of the reasons mentioned in the itemized list. Hence, to pro-mote such actions, we give a high positive reward. If the policy model plays a [FETCH] action, we give a small pos-itive reward. If GPT-4 rates the bot response as Bad for ei-ther of the actions, we provide a negative reward to demote such actions, e.g., the policy model gets a negative reward when it does not fetch context when required. To minimize the cost for GPT-4 evaluations, we only evaluate the cases where a policy model plays a [NO\_FETCH] action. This

5

where *γ* is the *discount factor* and *N* indicates the number of queries in a chat session. We set the value of *γ* to a small value (0.1) since we care for an immediate reward, i.e., ob-taining good output for each query.

We train the policy model on six chat sessions consist-ing of 168 queries. Additional chat sessions are created by randomly shuffling the queries within a chat session. For each query within a session, we sample a [FETCH] or [NO\_FETCH] action based on the initial policy. The sam-pled action is executed, and a bot response is generated. GPT-4 then evaluates the bot response and provides a rating as Good or Bad, which is then converted to a numeric re-ward. We repeat this process multiple times per chat session and create a dataset of 1733 (state, action, rating) tuples. As a policy model, we use an in-house pre-trained BERT model. The in-house BERT model has the same architec-ture as bert-base-uncased, with 12 layers and an encoding dimension 768. The model is trained using Masked language modeling (MLM) and Next sentence prediction (NSP) on in-domain text such as product descriptions, user reviews, etc. We add a linear layer with Softmax activation on the last layer’s embedding of the [CLS] token to map the State representation onto a 2-dimensional action space. We add two new tokens to the vocab representing policy actions, [FETCH] and [NO\_FETCH], and randomly initialize their embedding. The policy network is trained with the policy gradient loss and entropy regularization loss shown in Eq. 3. Entropy regularization is shown to help exploration and even leads to better optimization (Ahmed et al. 2019).

*lt* = *−* log *πθ*(*at|st*)*Gt − λ H*(*πθ*(*at|st*)) (3)

where *H* indicates entropy and *λ* indicates the entropy loss weight. We set *λ* to 0.1 for all our experiments. *st* indi-cates the state at *tth*time step and *at* indicates action taken at the *tth*time step.

We test the policy model on an extensive test chat session of 91 queries, which includes an interleaved sequence of in-domain queries (how to apply for card), greeting/acknowl-edgment queries (hi, ok, cool, thank u), and OOD queries. To judge the number of token savings obtained with the proposed optimizations, we compare the number of tokens passed to an LLM with a usual RAG pipeline to the settings where we introduce the optimizations for RAG. The number of tokens for LLM calls was calculated using the tiktoken library. With a usual RAG pipeline, we fetch a FAQ context for each query.

With a policy-based approach, the policy model provides a probability prediction over actions for each query. We use the Monte Carlo dropout method to estimate uncertainty and average the probabilities over ten predictions. A [FETCH] or [NO\_FETCH] action is selected based on the averaged probability.

**Similarity threshold on top-1 score (SimThr)**

As described earlier, the in-house embedding model trained with infoNCE loss provides accurate similarity scores for FAQs and discriminate scores w.r.t. OOD queries. An LLM call can be avoided if the top-1 similarity score is large.

6

|  |  |
| --- | --- |
| **query** | what cashback on i will get on swiggy |
| **policy action** | [FETCH] |
| **bot answer** | You will get 4% unlimited cashback on Swiggy with the credit card. |
| **human eval** | Good |
| **query** | what about pvr |
| **policy action** | [NO\_FETCH] |
| **bot answer** | You will get 4% unlimited cashback on PVR with the credit card. |
| **human eval** | Good |
| **query** | how much on curefit |
| **policy action** | [NO\_FETCH] |
| **bot answer** | You will get 4% unlimited cashback on Curefit with the credit card. |
| **human eval** | Good |

Table 9: Policy model predictions on a sequence of queries referring to the same FAQ

gpt-2, we use the embedding from the last layer’s [SEP] to-ken and add a linear layer on top of it. All the remaining experimental settings are the same as the in-house BERT experiment. Table 10 shows the comparison result. Using the gpt-2 model as the policy provides ~25% of token sav-ing with the same accuracy. This indicates that the in-house BERT model performs better with the limited training set due to domain pre-training.

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting** | **# tokens** | **saving** | **Acc.** |
| SimThr + policy (gpt-2) | 124924 | ~25% | 1. |
| SimThr + policy (BERT) | 115259 | ~31% | 1. |

Table 10: Comparison of token saving with a open gpt-2 as policy model

**Conclusion**

In this paper, we described a RAG-based chatbot for an-swering credit card-related queries using the domain FAQ dataset. We generated the question paraphrase data using the public LLM model and trained an in-house embedding model for retrieval using the infoNCE loss. An in-house model was shown to perform significantly better than a general-purpose public model in terms of ranking accuracy and OOD query detection. Further, we noticed that for spe-cific patterns/sequences of queries, it is not required to fetch the FAQ context to get a good answer. Assuming a fixed retrieval model and LLM, we optimize the number of to-kens passed to an LLM using Reinforcement Learning. We trained a policy model that resides external to the RAG and decides whether to retrieve a context. We used a GPT-4 as the reward model. The RL-based optimization combined with the similarity threshold led to significant token savings while slightly improving the accuracy. The proposed policy-based approach is generic and can be used with any existing

7