A Reinforcement-based QA Recommender System for Responding to Community-based Suggestions using Enhanced Contextualization

A Project Report

Presented to

The Faculty of the College of

Engineering

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science in Software Engineering

By

Abhishek Bais

Haley Feng

Princy Joy

Shannon Phu

December, 2021

APPROVED

Dr. Ali Arsanjani , Project Advisor

Prof. Dan Harkey, Director, MS Software Engineering

Prof. Dan Harkey, Department Chair

ABSTRACT

A Reinforcement-based QA Recommender System for Responding to Community-based Suggestions using Enhanced Contextualization

By Abhishek Bais, Haley Feng, Princy Joy, Shannon Phu

Community-based Question Answering (CBQA) platforms provide registered users a dedicated medium to interact with each other, find shared interest groups, post questions, crowdsource answers and learn from each other. In recent years, several CBQA platforms spanning verticals such as social news aggregation, business communication, retail, and social media platforms such as Pinterest, Instagram, and Facebook have gained traction.

These platforms provide users the ability to group and curate content. For example, Reddit, a social news aggregation platform, organizes its content into topic-based communities called subreddits. Users can join communities of interest by seeking help from other users on forums such as “r/FindAReddit”. However, with over 138,000 active communities finding the right community to join is challenging [1].

Recent work in CBQA recommender systems has focused on finding previously answered similar questions or community topic experts to make the best answer recommendations [2][3]. These approaches worked well for CBQA platforms when the number of questions was few. However, as the number of questions on these platforms grows, finding previously answered similar questions or community topic experts is not always possible.

In this project, we propose a Reinforcement-based Question-Answer Recommender (RQAR) system for responding to community-based network suggestions using enhanced contextualization. Our approach groups community-related questions on the Reddit platform by topic, type, and semantic similarity, extracts contextual information from questions, and utilizes user interactions to these questions to find, rank, and recommend relevant communities to users.

***Index Items* — Social Networks, Natural Language Processing, Machine Learning, Question Answering System, Text Classification, Semantic Similarity, Recommendation Systems, Reinforcement based learning, Sense and Contextualized embeddings.**

ACKNOWLEDGEMENTS

The authors are deeply indebted to Dr. Ali Arsanjani for his invaluable comments and assistance at every stage of this project. His extensive expertise in natural language processing and social network domains helped us build a deep understanding of the CBQA systems and their related challenges. His insights greatly helped us explore multiple schemes to find high-quality communities to the user's posted questions. We sincerely appreciate his constant encouragement and efforts to guide us in the right direction.

Table of Contents

List of Figures

List of Tables

Chapter 1. Project Overview

1.1 Introduction

In recent years, several Community-based Question Answering (CBQA) platforms spanning multiple verticals such as social news aggregation, business communication, retail, platforms for enthusiast programmers, and social media platforms for interactions and inquiry have gained traction. These platforms provide registered users access to a huge knowledge database of questions and answers accumulated over time. Users can post questions, seek crowdsourced answers, find shared interest groups, and learn from each other.

As CBQA platforms have gained popularity, the volume of questions and answers on them has increased exponentially. For example, in an interview with the Wall Street Journal in December 2020, Reddit, a leading social news aggregation platform, shared that “52 million people visit the site every day” [4]. Previously, Reddit shared that it had over 430 million monthly active users, who made 303.4 million posts in 2020, a 52.4% year-on-year increase. These posts received over 2 billion user comments, up 300 million from 2019 [5]. Year on year, CBQA platforms are growing in popularity and user base.

To help registered users find answers, several CBQA platforms organize their content in communities. Reddit organizes its content into topic-based communities called “subreddits”. Users can find relevant communities using tools such as “r/FindAReddit”

and join them to post questions and get community help on finding which subreddits to join. However, with over 2,620,000 subreddits and an average of 60,251 new subreddits added to Reddit each month, finding the right community to join is a challenging task [2].

To address this problem, a Reinforcement-based Question-Answer Recommender System (RQAR) is proposed to help CBQA users find relevant communities for their questions. This system will be used to extract context, group previously answered questions by similarity, learn from user interactions to find, rank, and recommend relevant communities to users.

1.2 Proposed Areas of Study and Academic Contribution

The proposed field of study is the factoid-based Question Answering (QA) task under Natural Language Processing (NLP) domain. The objective of factoid-based QA systems is to return short, single-word textual answers to user's queries. The proposed QA system focuses on answering a user's query about subreddit suggestions with a list of subreddits ranked by relevance. This QA task combines multiple disciplines of machine learning such as Information Retrieval (IR), Reinforcement Learning (RL), Ranking and Recommendation (RR), and Knowledge Graph (KG).

The IR unit of the system classifies the user's query into topics derived from the Latent Dirichlet Allocation (LDA) topic modeling and retrieves all historically answered similar question-answer pairs for that topic from the CBQA knowledge database. It uses a

cosine similarity test to calculate the semantic similarity between the user's query and the retrieved question-answer pairs. Question-Answer pairs with a high similarity score are used to prepare a list of candidate subreddits as answers. This list is passed to the RR unit where it is sorted based on a score calculated using the Learning to Rank (LTR) model. The KG is built using co-occurrence logic and maintains the subreddits and relationships between them as nodes and edges. The IR unit makes use of the KG features to recommend subreddits to the user as answers. The RL unit helps users formulate better questions to get more relevant subreddit recommendations.

The academic courses on machine learning and software engineering helped in forming strong foundations for this project. The Data Mining course introduced the fundamentals of machine learning and helped organize the life cycle aspects of this project like data preparation, model training, and model evaluation. The Web and Big Data Mining course provided a comprehensive understanding of large-scale analytics and encouraged undertaking a social media use case involving the collection and curation of large amounts of data. The Machine Learning course bolstered the understanding of different models on regression, classification, clustering, and NLP used extensively in this project. The course on Deep Learning introduced a new perspective on NLP with the transformer and BERT-based models critical to the success of this project. The Reinforcement Learning and Advanced Deep learning courses would help design multiple components in the system. Additionally, the Enterprise Software Platforms and

Software Systems courses helped plan, schedule, and build enterprise-grade software with continuous integration and continuous delivery.

1.3 Current State of the Art

Recent papers have introduced the current state-of-the-art (SOTA) technology for QA systems, text contextualization, and semantic similarity. Deep learning techniques using architectures such as Transformers and LSTMs have presented new ways of learning latent representations from sentences [2]. These techniques allow systems to compute semantic similarity between different QA pairs to rank different answers to questions [3].

It is also possible to treat the data source as a knowledge graph or network to find similarities between representations of objects in the network [6]. In terms of QA systems, Bidirectional Encoder Representations (BERT) has led to the latest SOTA performance. BERT can be applied to extractive QA tasks and used to provide contextualization and representation of natural language. These features are essential in finding accurate text similarities.

Chapter 2. Project Architecture

2.1 Introduction

The RQAR system architecture comprises three major components. These are the Paraphrase and Preprocessing Unit, the Information Retrieval Unit, and the Rank and

Recommendation Unit. The Paraphrase and Preprocessing Unit collects the input data and uses reinforcement learning to refine user questions in order to get a more well-written question prior to information retrieval. Data cleaning and distillation is also performed to prepare the input data for semantic similarity in the next stage. The Information Retrieval Unit is the main logic of the system. It consists of generating embedding and computing semantic similarity to provide candidate subreddits as answers to user-posted questions. The Rank and Recommendation Unit ranks the candidate subreddits to make relevant community recommendations to the user. Figure 1 shows the system architecture.

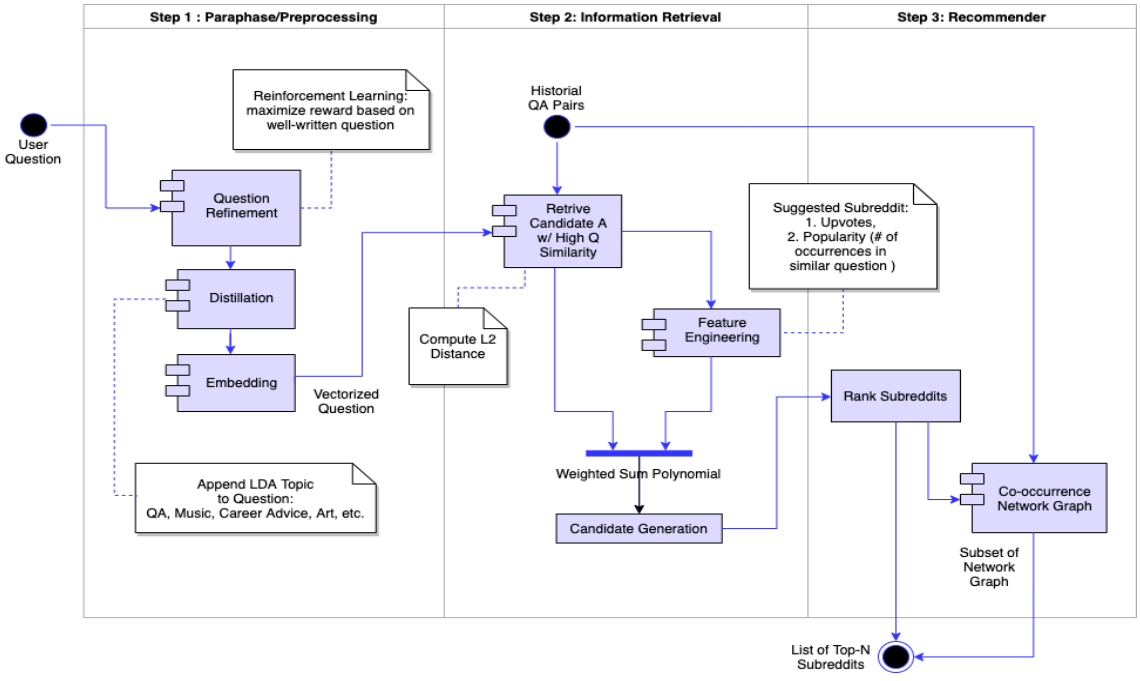

Figure 1. System architecture

2.2 Architecture Subsystems

2.2.1 Paraphrase and Preprocessing Unit

The Paraphrase and Preprocessing Unit collects a user posted question and proposes a paraphrase back to the user using Reinforcement Learning. The Question Refinement module uses a policy and a rewards function aimed at maximizing the reward for the well-formed question. It does so using the Microsoft Azure 'Personalizer' service and Parrot paraphraser utility built on top of the Text to Text Transfer Transformer (T5). These rewards are used by the system to generate refined questions that are provided to the user to seek feedback and improve subsequent queries.

If the user agrees to the paraphrase questions, the question is fed into the topic modeling API to categorize the question into a known topic. Here, the input will have the topic keywords append to the end for extra contextual information prior to embedding. A pretrained sentence transformer model will encode the question into a semantic representation and pass this embedding to the Information Retrieval Unit for generating candidate subreddits as answers.

2.2.2 Information Retrieval Unit

The Information Retrieval Unit consists of a QA database generated from Reddit scraping APIs such as PRAW and Pushshift. A candidate answer retrieval API calculates the semantic similarity between the user's question and all historical questions using Euclidean distance. Historical questions with high semantic similarity are used to query related subreddits and their upvotes. Additional features such as popularity of the subreddits among related questions are created during this process. The semantic similarity and additional features of these subreddits are used to compute a final weighted score. The subreddits and their respective scores are passed to the Rank and Recommend Unit to make relevant community recommendations to the user.

2.2.3 Rank and Recommend Unit

The Rank and Recommend Unit ranks the candidate answers generated by the Information Retrieval Unit by total relevance weight to make available high-quality community recommendations to the user as answers. A separate co-occurrence graph API builds a network knowledge graph using all subreddits from the QA database. A subset of the network graph will be displayed along with the recommendation to show the relationship between the final top N subreddits.

Chapter 3. Technology Descriptions

3.1 Data-Tier Technologies

3.1.1 Reddit Pushshift and PRAW

Pushshift provides an API to query the Reddit CBQA forum for historically answered questions for a given date range. The output of a Pushshift request is a JSON file that can be filtered by subreddit answers, searched for user upvotes, authors, etc. Similar to Pushshift, PRAW also provides APIs to query the Reddit CBQA forum for historically answered questions for a given date range.

3.1.2 Pandas, NLTK, and Gensim

Pandas is a popular open-sourced data analysis and manipulation Python library used throughout our implementation process for reading and writing dataframes. The powerful library allows us to extract user suggested subreddit from comments, create new features from existing features, and perform normalization techniques to scale our data.

Natural Language Toolkit (NLTK) is an NLP Python library commonly used to preprocess text data. This module is used during our data cleaning stage to remove unnecessary words and duplications, shorten words to their root form, and expand contractions.

Topic Modeling is performed using Python's Gensim package. A Gensim LDAMallet model is trained to recognize hidden topics from the entire question corpus. In order to find the most optimal number of topics, the model is evaluated based on

Gensim's coherence score. The model with the highest coherence score is picked to perform topic modeling on our dataset.

3.1.3 Networkx and Bokeh

Boheh is a data visualization Python library used to create interactive graphs. We integrated this package with the Networkx library for studying graphs and networks to create a network graph showcasing subreddits co-occurrence relationships.

3.2 Middle-Tier Technologies

3.2.1 TF-IDF and Sentence Transformers

For transforming text into vectorized form, we experimented with three word encoding techniques: Term Frequency-Inverse Document Frequency (TF-IDF), Sentence Transformers pretrained with BERT, and Sentence Transformers pretrained with Multi-QA-MiniLM.

We experimented with Sklearn's TFidfVectorizer to transform questions into numerical representation by measuring the weight and significance of each word in the entire question corpus.

Sentence Transformers is a Python framework for SOTA sentence embedding with a large collection of pre-trained models designed for various text-related tasks. For our implementation, we tried text embedding with a regular BERT model that maps each question to a 768 dimensional dense vector appropriate for semantic search. Another sentence transformer model we used is the Multi-QA-MiniLM that maps each question to

a 384 dimensional dense vector. This model was trained on 215M question-answer pairs from a diverse group of datasets including WikiAnswers, Stack Exchange, and Amazon-QA.

3.2.2 Faiss

Faiss is a Python library developed by Facebook AI Research to efficiently compute similarity search for a large text corpus. The framework takes in the input of an embedded question vector with meaningful representation and searches for similar historical questions by computing L2 (Euclidean) distance. Historical questions with the lowest L2 distance are returned.

3.2.3 Azure Personalizer and Parrot

Azure Personalizer is an Azure cloud cognitive service that provides an online learning platform to implement reinforcement-learning applications out of the box in the form of an easy-to-use API. Our system extracts online user interactions from our website and submits them to the Personalizer service to learn refined user questions.

The other technology we use is a Parrot Paraphraser. Parrot Paraphraser is an open-source Python library that is based on a T5 natural language model that generates different paraphrases for a given input sentence. The RQAR system seeks user input to configure the paraphraser to return paraphrased responses that preserve meaning

adequately, are fluent in English or are as close as possible to the original sentence lexically and syntactically.

3.3 Client Technologies

3.3.1 Streamlit

Streamlit is a Python UI framework designed specifically for building data science applications. Streamlit makes it easy for machine learning engineers to build powerful web applications in just a few lines of code without prior experience in frontend design. The app is a simple Python script that runs from top to bottom without using complex callbacks that are typical of most UI frameworks. It has a wide range of customizable widgets for user interaction, supports multiple python data science and visualization libraries that are the core components of any machine learning application. Streamlit offers effortless installation, configuration and deployment of the application.

3.3.2 Google Cloud and Kubernetes

Google Cloud Platform offers a variety of services that can be tailored to the needs of an application. We take advantage of the GitOps-style Continuous Integration and Continuous delivery (CI/CD) capability for automating our deployment cycles. We use Google Cloud Build to execute builds and create container images for our application. The committed code can be built and tested automatically using triggers in Github repositories. Our container images are stored and managed by the Google Container Registry. We use Google Kubernetes Engine (GKE), a managed Kubernetes

cluster service to host our containerised application. These managed clusters eliminate the operational overhead using autoscaling. In Kubernetes, applications are represented as Pods, which are scalable containers. These pods can distribute and scale to meet users' needs with a load-balanced set of replicas. To expose the cluster and make it available through the public internet we use External Load Balancer IP.

Chapter 4. Project Design

4.1 Data-Tier Design

After collecting Reddit data using Pushshift and PRAW APIs, the question and comment dataframes are created. Questions and descriptions are combined and preprocessed using the text cleaning API to create a new QuestVocab feature.

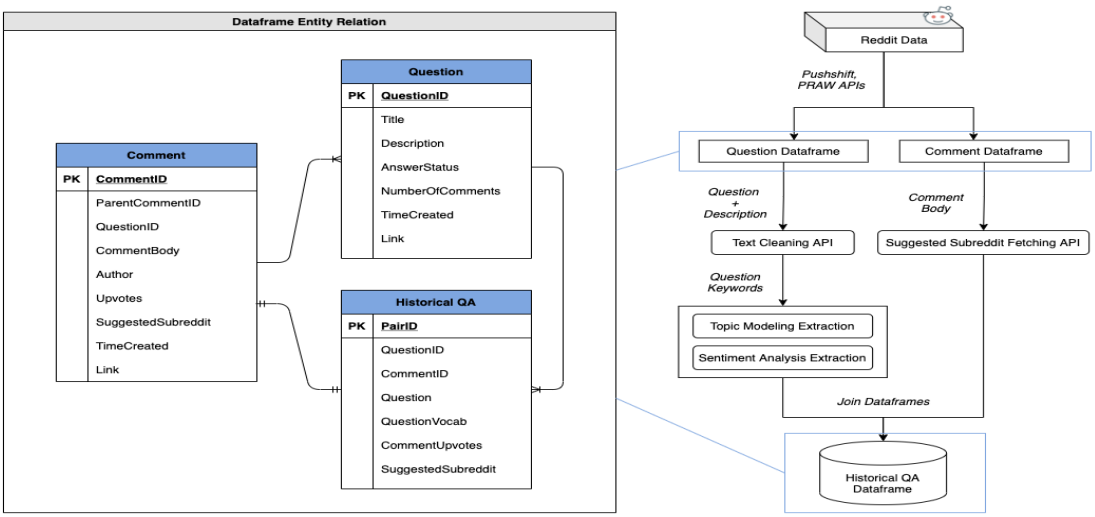

Figure 2. Historical QA Dataframe Configuration

Next, topic modeling and sentiment is performed on the preprocessed question to extract interesting features. Upvotes of subreddit answers are also added to the comment dataframe. Finally, the two dataframes are merged to create the historical question answering dataframe.

Each question in the dataframe can have more than one suggested subreddit, thus projecting a one-to-many relationship between the question dataframe and the historical QA dataframe as shown in the Historical (Question-Answering) QA configuration diagram in Figure 2.

4.2 Middle-Tier Design

4.2.1 Information Retrieval

The Information Retrieval component provides the main logic of the RQAR system. It is responsible for finding high-quality candidate subreddit answers to the user's question. It does so using five key sub-components namely the Historical Question-Answer Handler, the Semantic Similarity Handler, the Social Interaction Handler, the Knowledge Graph, and a Candidate Answer Generator.

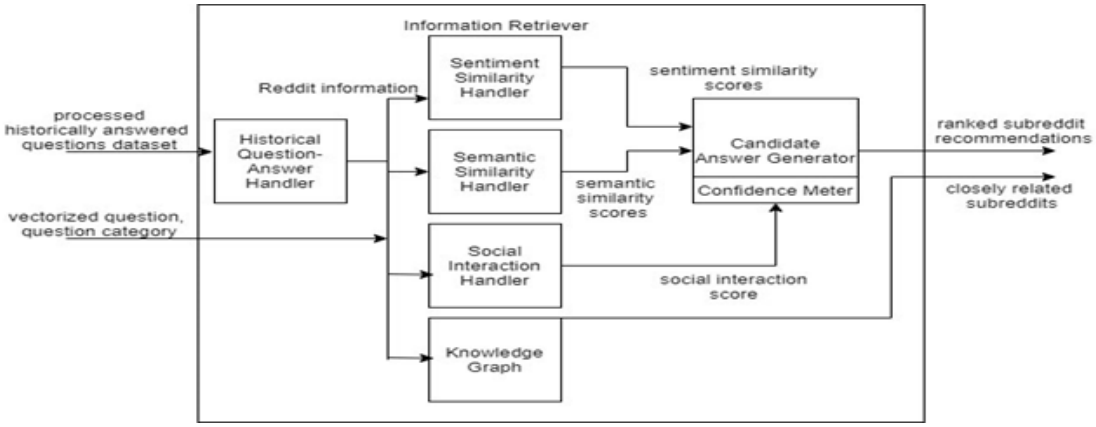

Figure 3. Data Flow diagram for the Information Retriever

In addition, this component also creates and maintains a Candidate-Answer database of candidate subreddit answers and a Co-occurring Answer database of subreddits that are closely related to each other to provide to the Recommendation Handler for further processing. Figure 3. shows the data flow diagram for the component.

To find the best methodology for Information Retrieval leading to high-quality recommendations, this component creates two machine learning pipelines. The first is based on using user feedback as sentiments to historically answered questions and the second is based on findings semantically similar historically answered questions to the user's question for the Reddit platform. Figure 4. shows a pipeline that uses a sentiment and social interaction score-based scheme to make ranked subreddit recommendations.

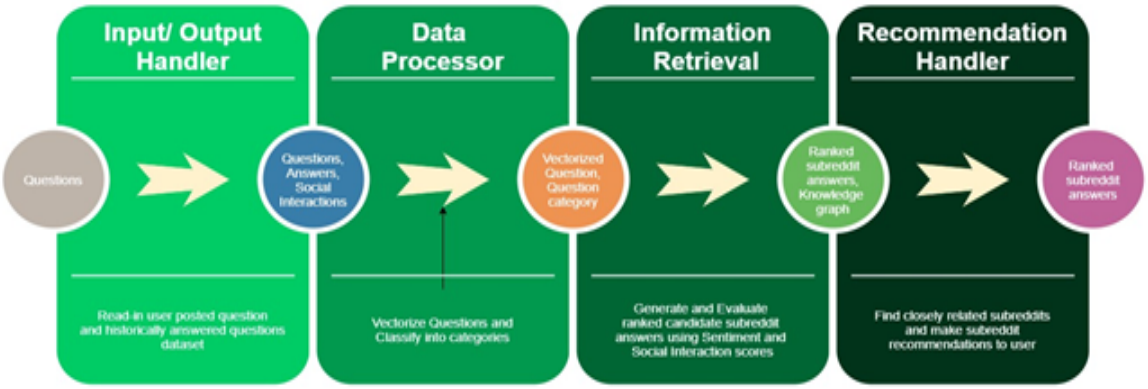

Figure 4. Machine Learning pipelines using Sentiment and Social Interactions

Figure 5. shows a pipeline that uses a semantic and social interaction score-based scheme to make ranked subreddit recommendations.

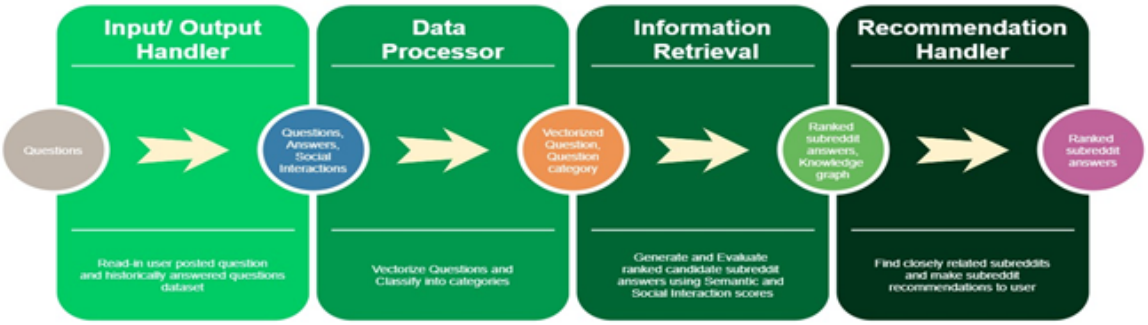

Figure 5. Machine Learning pipelines using Semantic and Social Interactions

4.2.2 Reinforcement Guided Question Refinement

The Reinforcement Guided Question Refinement component uses the Text to Text Transfer Transformer (T5), a Parrot Paraphraser and the Microsoft Azure Personalizer service to generate refined questions based on the users' question refinement type

selected.

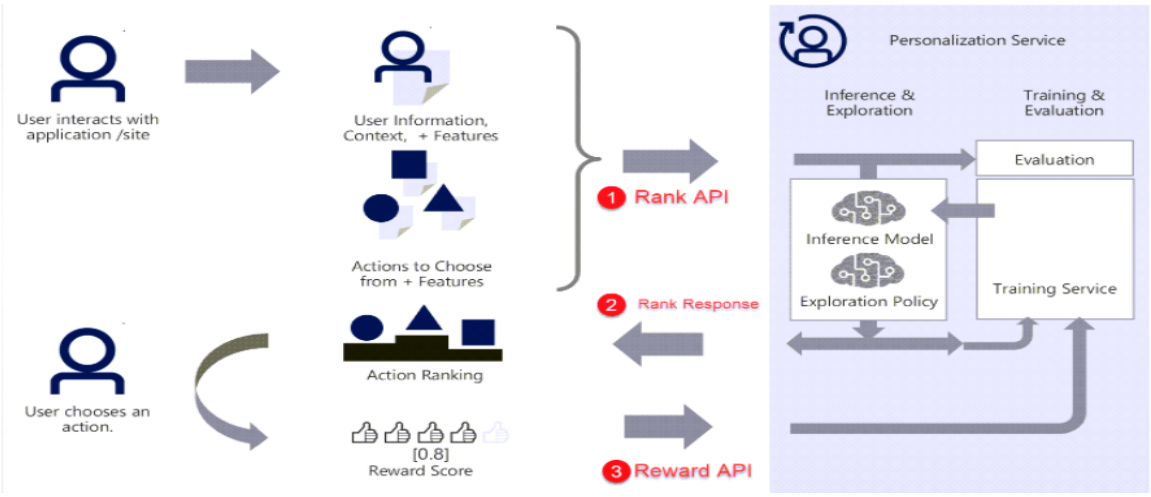

Figure 6. Architectural flow diagram of calling the Microsoft Azure Personalizer's Rank and Reward

First, T5 is used to generate five refined question choices using Parrot as an augmentation utility. Next, these five choices are passed as *context features* to the Personalizer's Rank API. Then, the Personalizer ranks the refined question choices and returns the top-ranked choice to the RQAR system as a reward. The system presents the top-ranked choice to the user for feedback. Positive user feedback results in the RQAR system generating a reward score that is registered with the Personalizer for future question refinement requests. Figure 6. shows the architectural flow diagram of the RQAR system's interaction with the Personalizer.

4.3 Client Design

4.3.1 Streamlit Web Interface

The user interface of the web application is designed to have two pages.

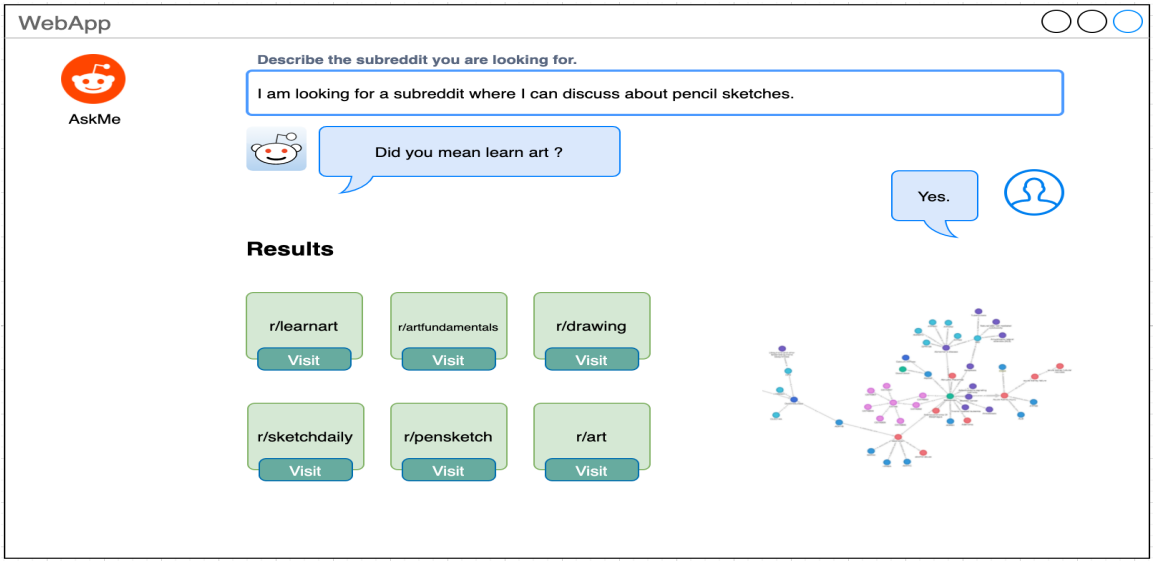

Figure 7. Mockup Design of the Homepage of the RQAR web application

The home page is shown in Figure 7. It facilitates the main functionality of the RQAR system. It provides an interface to accept user questions and display ranked recommendation results. It also provides an interface to configure the system for reinforcement learning guided question refinement using the parrot paraphraser to return paraphrased questions that preserve meaning adequately, are fluent in English, or are as close as possible to the original sentence lexically and syntactically.

The second page is shown in Figure 8. It presents the knowledge graph showing the underlying relations between the subreddit communities.

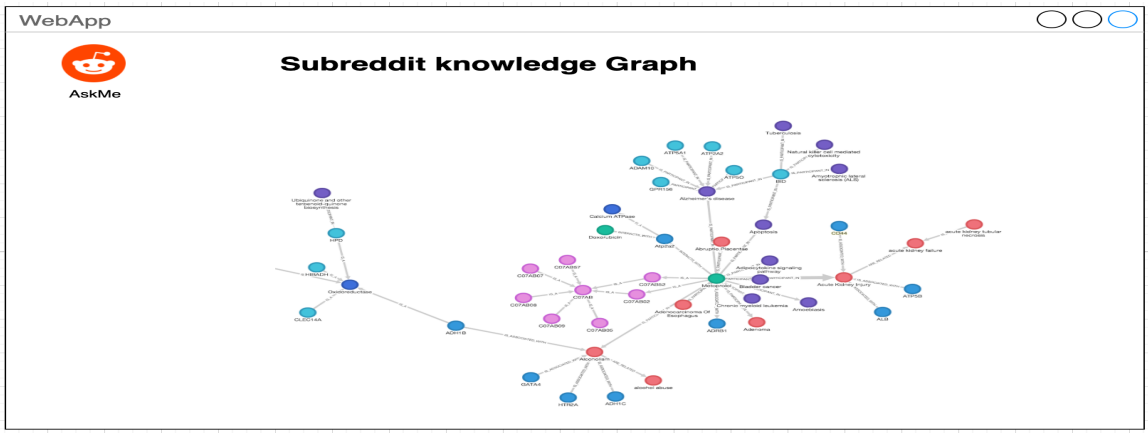

Figure 8. Mockup Design for the second (knowledge graph) page

The Input/ Output Handler uses Streamlit, a python library that facilitates rapid development of machine learning web applications.

Chapter 5. Project Implementation

5.1. Input/ Output Handler

This component acts as the sole entry and exit point of the RQAR system. It uses Streamlit to provide Reddit users a web interface with a textbox that accepts questions to which they seek ranked subreddit recommendations as answers. The user questions are served to the downstream Data Processor component described in section 5.2 for further processing. Once the ranked subreddit recommendations are available by the Recommendation Handler described in section 5.4.1, the Input/ Output Handler provides a web interface to display the results.

In addition, this component provides Team Arsanjani-Ali Su21-1 developers an interface to read in a dataset of historically answered Reddit questions. These questions form the knowledge corpus for the Information Retrieval component to generate high-quality candidate and related subreddits as answers and added guidance respectively.

Pushshift and PRAW APIs are used to prepare the historically answered questions dataset. Figure 9. provides an example Reddit Pushshift request while Figure 10. provides an example Reddit PRAW request. A combination of Pushshift and PRAW schemes is used to avoid hitting the Reddit API rate limit of 30 requests per minute to prepare a statistically significant corpus of historically answered questions for data mining tasks performed by the Information Retriever component.

```
import requests
query="seo" #Define Your Query
url = f"https://api.pushshift.io/reddit/search/comment/?q={query}"
request = requests.get(url)
json_response = request.json()
json_response
```

Figure 9. An example Reddit Pushshift request to retrieve Reddit data

```
import praw

reddit = praw.Reddit(client_id='clientid',
                     client_secret='secret', password='password',
                     user_agent='PrawTut', username='username')

subreddit = reddit.subreddit('python')
```

Figure 10. An example PRAW request to retrieve Reddit data

5.2 Data Preprocessing Unit

This component prepares the user-provided and historically answered questions for information retrieval. It does so by running data cleaning tasks such as removing punctuations, links, and stop words, expanding word contractions, and performing word lemmatization on the questions. The processed questions are curated into a dataset and passed on to the Question Category classifier and Embeddings/ Vectorizer described in section 5.2.1 and 5.2.2 to prepare for downstream information retrieval tasks. The Data Processor employs a subset of data cleaning tasks described in Angiani et al. [8] to curate the data. Table 1.0 provides a full list of the data cleaning tasks performed.

Raw question content	**Description/Examples**	**Cleaning Technique**
Non-English, Incomplete	Trailing or leading white spaces, Numbers, Duplicated Word, Not in the English Dictionary	Remove
Links/ URLs	http, com, ww	Remove
Punctuations	./:-!?$%	Remove

Word Contractions	ain't: is not, aren't: are not, can't: cannot	Replace with expanded form
Unnecessary capitalizations	"Looking for a" : "looking for a"	Replace with lowercase
Contentless stopwords	- Reddit-related: subreddit, subreddits, reddit, sub, community - Helper words: question, like, post, find, help, want, look, ask, people, something, talk	Stop words removal
Inconsistent word form	better: good, corpora: corpus	Lemmatize

Table 1: Data cleaning tasks performed by the RQAR system before information retrieval

The cleaned questions are then passed to the Embedding/ Vectorizer and the Question Category Classifier described in sections 5.2.1 and 5.2.2 respectively for further pre-processing.

5.2.1 Question Category Classifier

This sub-component accepts a question as input and classifies it into topics discovered from topic modeling. The historical QA database is used to train a LDAMallet model and the model with the highest coherence score is pickled for use during inferencing. The user-provided question is fed into the LDA inference model to compute LDA scores that indicate the probability of the question belonging to a specific topic. The topic with the highest LDA score for the input question is used by appending the keywords associated with the topic to the end of the question. Figure 11. shows the highest weighted keywords for each topic discovered using LDA and their distribution

while Figure 12. shows the LDA Topic Modelling Process.

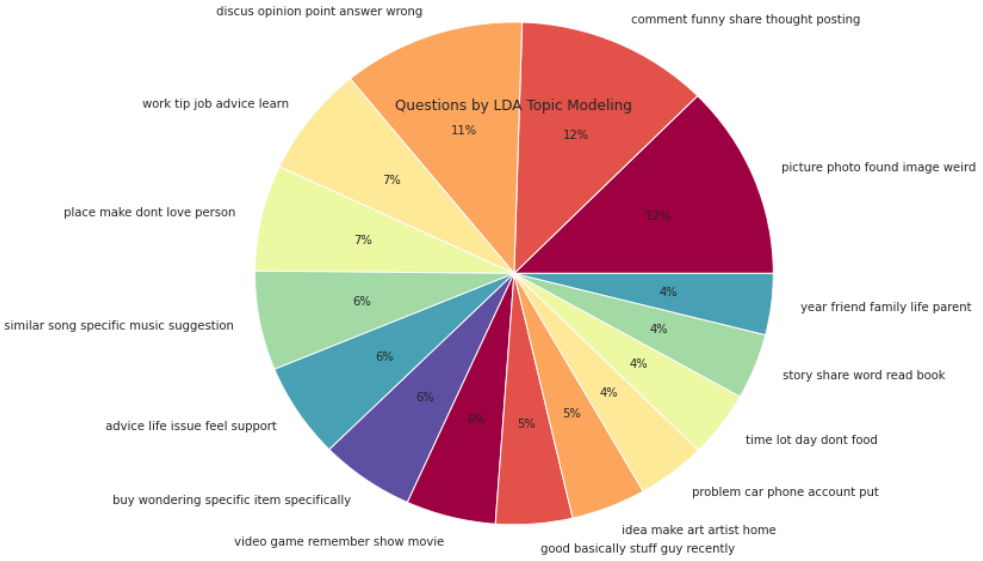

Figure 11. Keywords for each topic discovered using LDA and their distribution

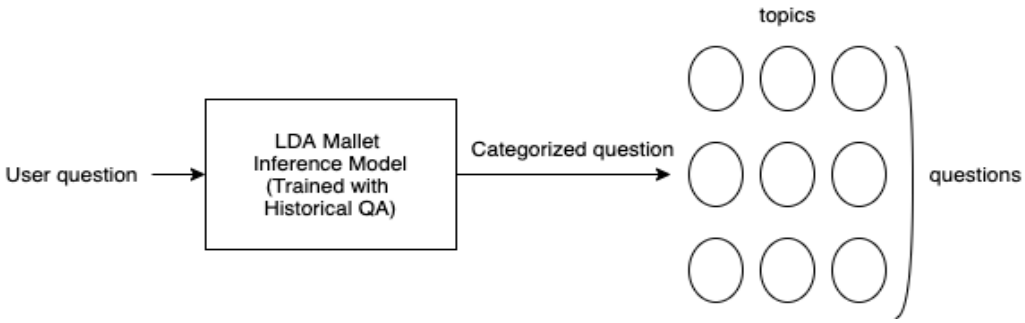

Figure 12. LDA Topic Modelling Process

Topic Modelling allows the system to make intelligent semantically similar searches on the historically answered questions. It does so by adding additional contextualized information to the question prior to embedding. The Data Processor passes both the vectorized question and its category classification to the Information Retriever

component for further processing. Figure 13. shows the data flow through the Data Processor.

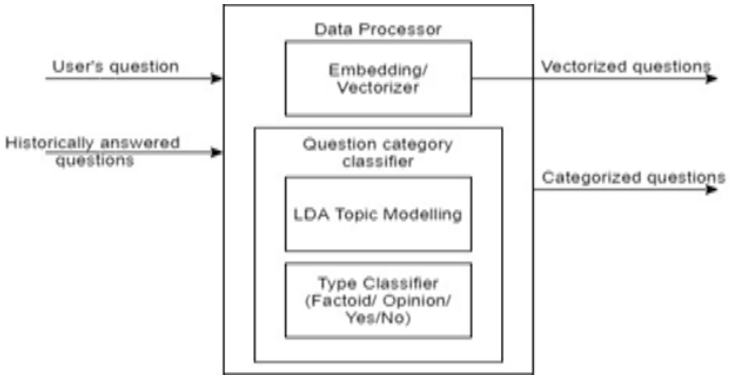

Figure 13. Data flow in and out of the Data Processor

5.2.2 Embeddings/Vectorizer

This sub-component uses word embeddings to convert cleaned questions to a vectorized form with semantic value. Each question is encoded into a 384-dimensional dense vector using the sentence transformers model Multi-QA-MiniLM. This sentence transformers model was pre-trained on 215M question-answer pairs for computing semantic similarity.

5.3 Information Retrieval Unit

This component contains the main logic of the RQAR system. It is responsible for finding high-quality candidate subreddit answers to the user's question. It does so using six key sub-components namely the Historical Question-Answer Handler, Semantic Similarity Handler, Social Interaction Handler, Knowledge Graph, Candidate Answer

Generator. These sub-components are described in sections 5.3.1 through 5.3.5. Figure 14. shows the sequence diagram of the Information Retriever.

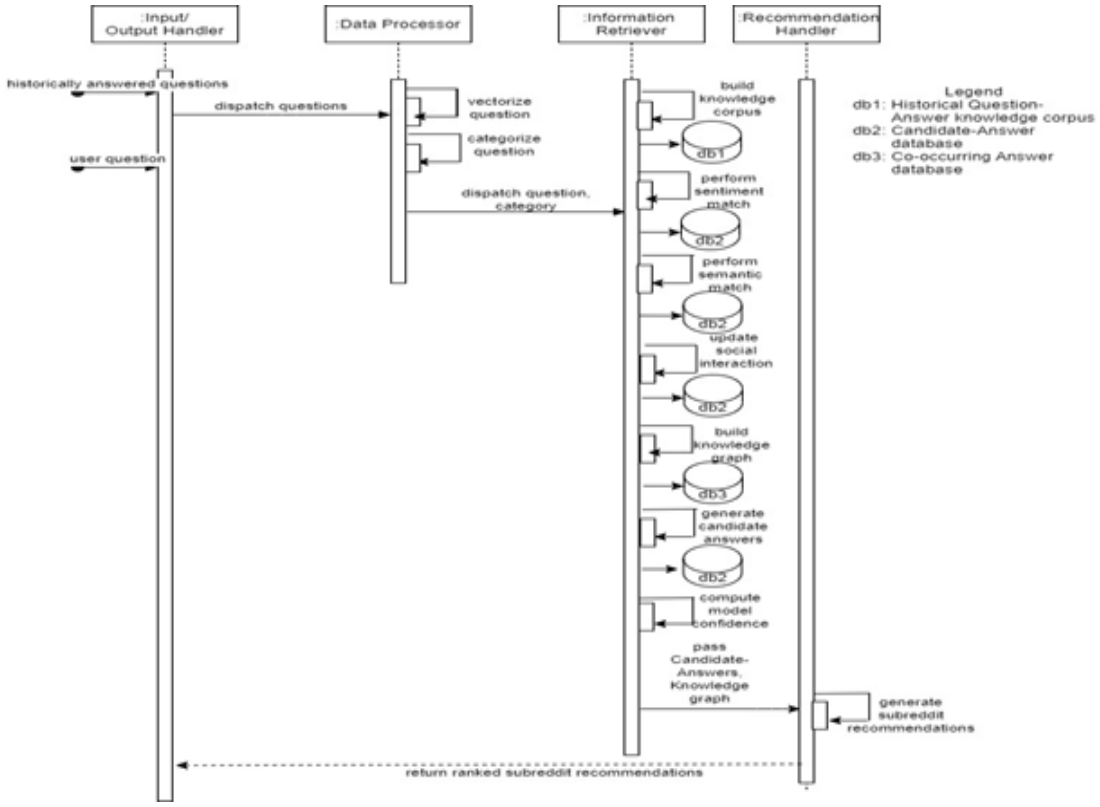

Figure 14. Sequence diagram of the Information Retriever component

In addition, this component also creates and maintains a Candidate-Answer database of candidate subreddit answers and a Co-occurring Answer database of subreddits that are closely related to each other to provide to the Recommendation Handler for further processing.

5.3.1 Historically Answered Questions Handler

This sub-component accepts the historically answered Reddit questions dataset from the Input/Output Handler and processes it to build a knowledge corpus of question-answers that the other sub-components of the Information Retriever component query to perform data mining tasks.

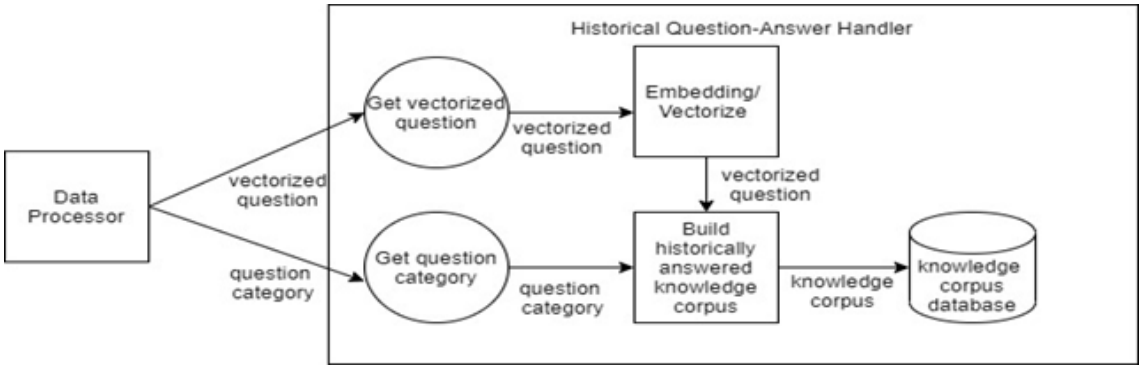

Figure 15. Handshake between Data Processor and Historical Question-Answer Handler

Figure 15. shows the handshake between the Data Processor and the Historical Question-Answer Handler components to get vectorized questions, categorize classification by topic, and build a knowledge corpus of historically answered questions with subreddit answers and user upvotes.

5.3.2 Sentiment Similarity Handler

This sub-component accepts the user question in its vectorized form from the Data Preprocessing Unit and performs a sentiment similarity match with each historically answered question in the knowledge corpus obtained from the Historical

Question-Answer Handler. Each historically answered question for the user question's topic is saved as a candidate subreddit answer in the Candidate-Answer database with its associated sentiment similarity score. Table 2. shows some historically answered questions scored by sentiment.

	question_vocab	comment_upvotes	suggested_subreddits	LDA_topic	LDA_topic_score	VADAR	compound	sentiment_type
7331	show peopl good ear piec music recognis note t...	3	r/musictheory	23	0.528358	{'neg': 0.0, 'neu': 0.649, 'pos': 0.351, 'comp...	0.6597	1
8719	post pictur cloth seen somewher onlin shirt pa...	10	r/findfashion	24	0.702470	{'neg': 0.042, 'neu': 0.808, 'pos': 0.15, 'com...	0.5574	1
12703	dedic histori war memorabilia artifact	1	r/germanmilitaria	23	0.252389	{'neg': 0.494, 'neu': 0.506, 'pos': 0.0, 'comp...	-0.5994	0
692	hotel employe help hotel employe ask system re...	2	r/talesfromthefrontdesk	16	0.340770	{'neg': 0.0, 'neu': 0.698, 'pos': 0.302, 'comp...	0.6369	1
21796	like vexillolog currenc coin bank note like ve...	1	r/coins	19	0.090561	{'neg': 0.192, 'neu': 0.588, 'pos': 0.22, 'com...	-0.0258	0

Table 2. Sentiment analysis on some historically answered questions

5.3.3 Semantic Similarity Handler

This sub-component accepts the user question in its vectorized form from the Data Processing Unit and performs a semantic similarity match with each historically answered question in the knowledge corpus obtained from the Historical Question-Answer Handler. Each historically answered question for the user question's topic is saved as a candidate subreddit answer in the Candidate-Answer database with its associated semantic similarity score. Figure 16. shows the semantic analysis process.

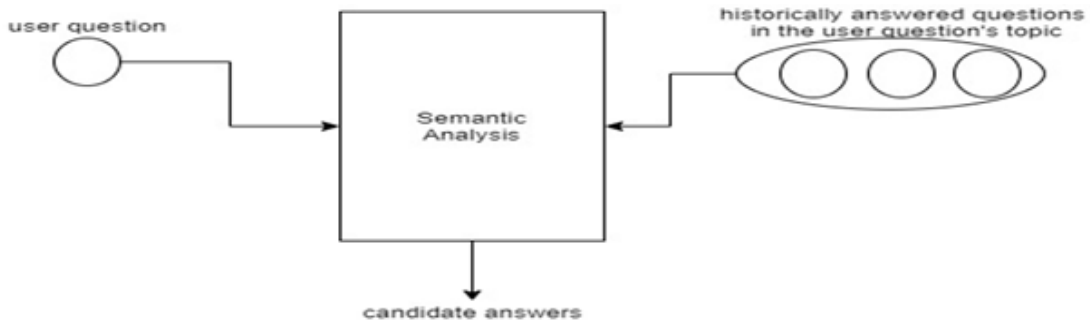

Figure 16. Semantic Analysis process

5.3.4 Social Interaction Handler

This sub-component goes over each candidate subreddit answer in the Candidate-Answer database and queries the knowledge corpus for the number of upvotes the answer received from Reddit users. It adds this information as a social interaction score to the candidate subreddit answer in the Candidate-Answer database. Table 3. shows some historically answered questions scored by semantic similarity.

	question_vocab	comment_upvotes	suggested_subreddits	LDA_topic	LDA_topic_score	best_semantic_match_subreddits	best_semantic_match_score	semantic_type
22965	peopl post small wood work project like tini w...	1	r/woodworking	24	0.213166	r/findfashion	0.890395	1
23714	music post arrang peopl link music sheet amate...	1	r/sheetmusic	23	0.256016	r/findfashion	0.902783	1
11179	peopl show funni way break rule someth logic w...	1	r/madlads	15	0.108624	r/redditinreddit	0.888034	1
26062	find specif video pictur describ	2	r/helpmefind	24	0.808000	r/redditinreddit	0.842851	1
6341	kind brag skill non physic achiev freeli ya bo...	1	r/congratslikeimfive	22	0.136932	r/fancyfollicles	0.803180	1
24155	favor reopen societi	6	r/anarchocapitalism	18	0.010000	r/breadit	0.781435	0

Table 3. Semantic Analysis on some historically answered questions to the user's question

semantic_type = 1 if match > 80%

semantic_type = 0 if match < 80%

5.3.5 Candidate Answer Generation Handler

This sub-component generates ranked subreddit answers using the information provided by the Sentiment Similarity, Semantic Similarity, and Social Interaction handlers.

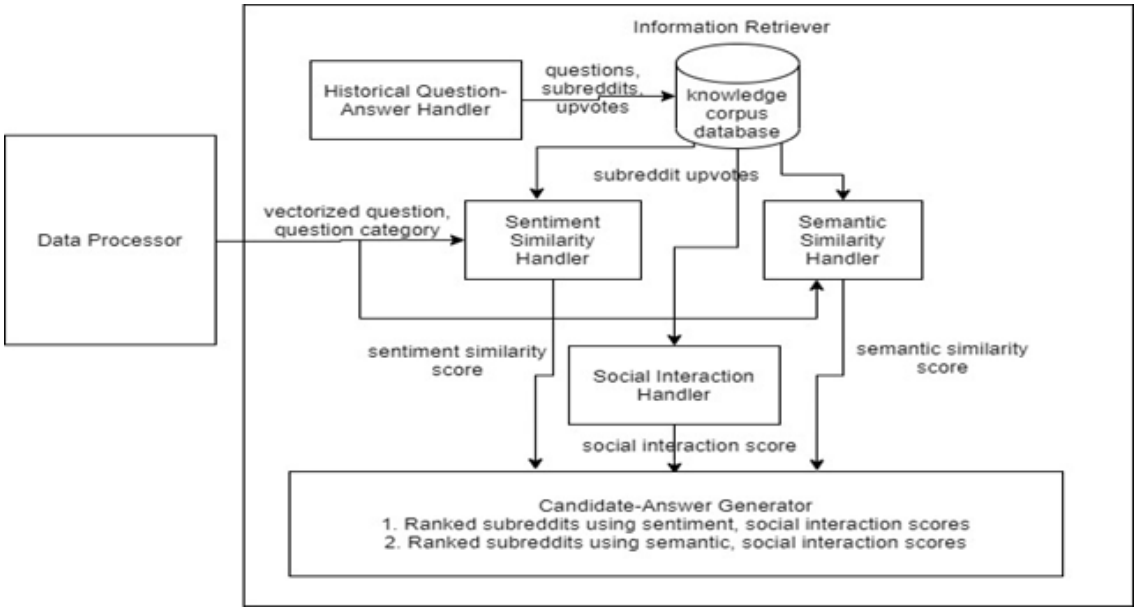

Figure 17. The Candidate-Answer generation process

Figure 17. shows the Candidate-Answer generation process that is used to prepare two different lists of ranked subreddit answers. The first list is prepared by ranking subreddit answers using the sentiment similarity score computed by the Sentiment Similarity Handler described in section 5.3.2 while the second list will be prepared by ranking subreddit answers using the semantic similarity score computed by the Semantic Similarity Handler described in section 5.3.3. Tie-breaks in sentiment or semantic

similarity scores are resolved using the social interaction score computed by the Social Interaction Handler described in section 5.3.4. Both lists of ranked subreddit answers are added to the Candidate-Answer database for the user's posted question and passed on to the Recommendation Handler to prepare subreddit recommendations.

5.4 Recommendation Unit

5.4.1 Rank and Recommendation Handler

This component collates the ranked subreddit answers contained in the Candidate-Answer database and closely related subreddits to them contained in the Co-occurring Answer database as shown in Figure 18. to provide ranked and related subreddit recommendations to the Input/ Output Handler for relay to the user.

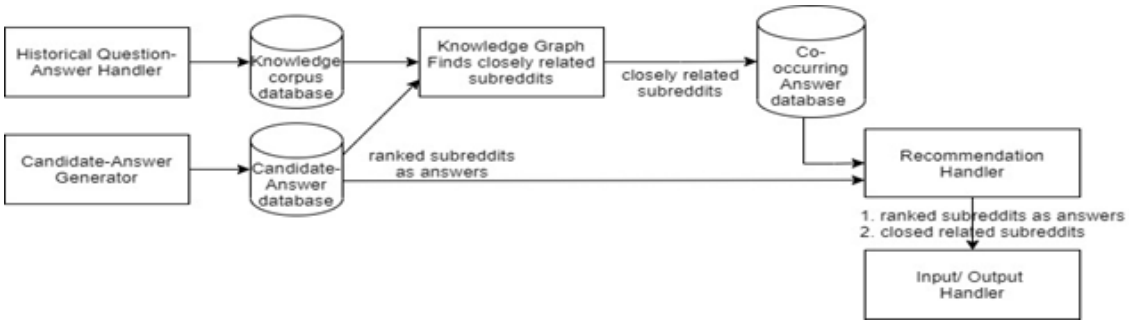

Figure 18. Recommendation Handler - subreddit recommendations

5.4.2 Knowledge Graph Generation

This sub-component first iterates over the knowledge corpus and performs KNN clustering on it to build a Co-occurring Answer database of subreddits that are closely

related to each other. Next, for each candidate subreddit answer in the Candidate-Answer database, it finds the cluster of subreddits most closely related to it. Finally, the closely related subreddits are passed on to the Recommendation Handler to generate high-quality candidate and related subreddits as answers and added guidance respectively for the user.

5.5 Reinforcement Guided Question Refinement Unit

The Reinforcement guided question refinement process is a two-step process namely Question Refinement by paraphrasing and Adaptive Reinforcement Learning. These are described in greater detail in sections 5.5.1 and 5.5.2.

5.5.1 Question Refinement

This sub-component refines the user questions based on the user selected question refinement type. To do so, it uses the Text to Text Transfer Transformer (T5) and the Parrot paraphraser. The T5 is a text-to-text framework that allows use of the same model, loss function, and hyperparameters for any NLP task.

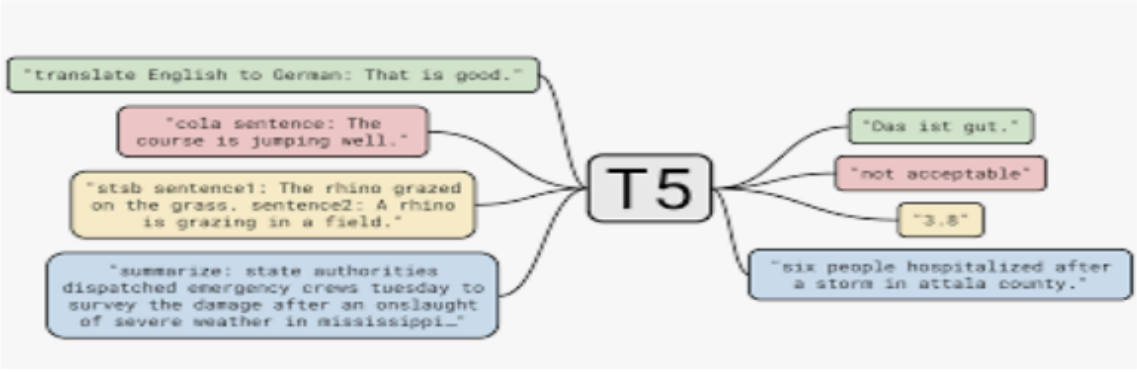

Figure 19. Diagram of the T5

Figure 19 shows a diagram of the T5 framework. It accepts the user's question as text input to the model, which is trained to generate the target refined question text.

The parrot paraphraser is used as a text augmentation engine to T5. It provides 3 different types of text augmentation, namely Adequacy- "Is the meaning preserved adequately?", Fluency - "Is the question text in fluent English?" and Diversity - "Is the refined question lexically and syntactically close to the original question?" The augmented technique is passed to the T5 which generates refined question choices. Figure 20. shows an example question and its refined questions.

```
----------------------------------------------------------------------
Input_phrase: Can you recommed some upscale restaurants in Newyork?
----------------------------------------------------------------------
list some excellent restaurants to visit in new york city?
what upscale restaurants do you recommend in new york?
i want to try some upscale restaurants in new york?
recommend some upscale restaurants in newyork?
can you recommend some high end restaurants in newyork?
can you recommend some upscale restaurants in new york?
can you recommend some upscale restaurants in newyork?
----------------------------------------------------------------------
```

Figure 20. Question and list of refined questions

5.5.2 Adaptive Reinforcement Learning

The adaptive reinforcement learning of refined questions takes place using the Microsoft Azure Personalizer service. It provides a Rank API to select the best query (*action*) based reward scored for the user. The RQAR system feeds the top five, T5 generated refined queries as *context features* to the Personalizer's Rank API which ranks

them and returns the most refined question to the user for feedback. Based on the feedback provided, the Personalizer generates a reward and uses it to learn refined questions to user questions. Figure 21. shows the data flow diagram of the RQAR system's interaction via the user with the Personalizer service.

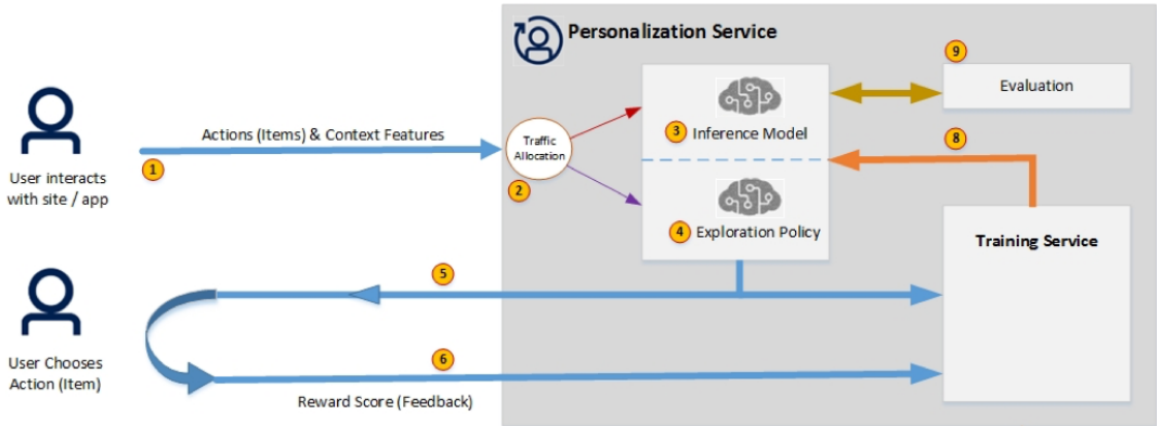

Figure 21. RQAR system's interaction via the user with the Personalizer service.

5.6 Streamlit Web Application

The Streamlit web interface consists of two pages. Figure 22. and Figure 23. show the Home Page that facilitates the user's interaction with the RQAR system. It provides the user navigational help and directions. The main widget is the text box where the user types-in their search question. The user gets paraphrased suggestions for their question based on the preferences they set on the advanced environment menu. The user can accept or reject the paraphrased query.

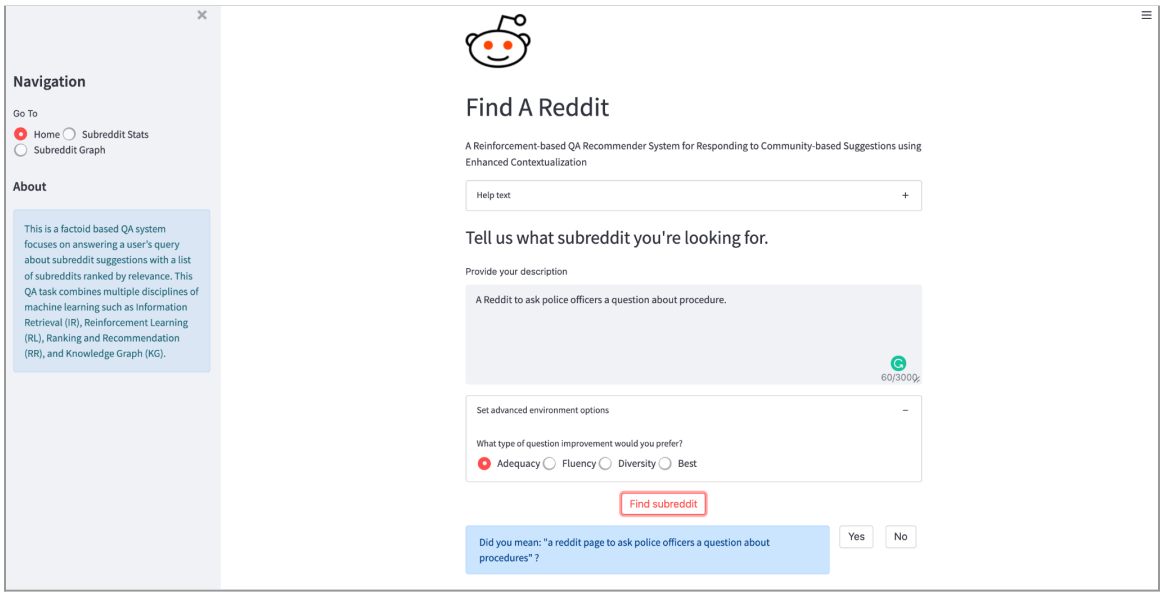

Figure 22. Home page of the web application which shows the textbox widget to type user query

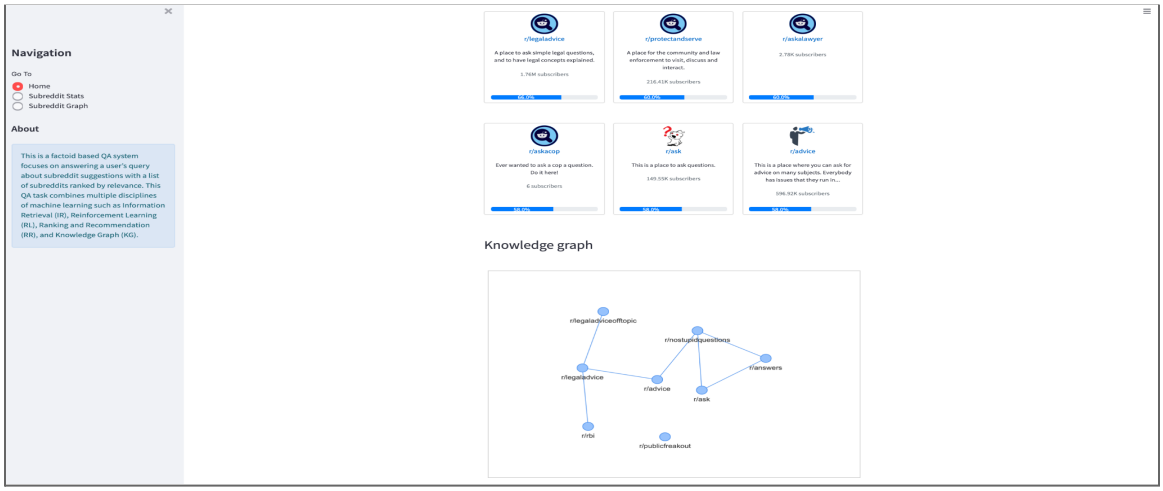

Figure 23. Home page showing the recommended subreddits for the user query

The results section on the Home Page shows a list of recommended subreddits in ranked order. Each recommendation includes a short description, subscriber count, and a confidence bar that shows how confident the RQAR system is in making the subreddit recommendation. A knowledge graph is also generated to show how the recommended

subreddits are related to each other. Figure 24 shows the second (knowledge graph) page that provides an interactive knowledge graph displaying the organization of subreddit communities.

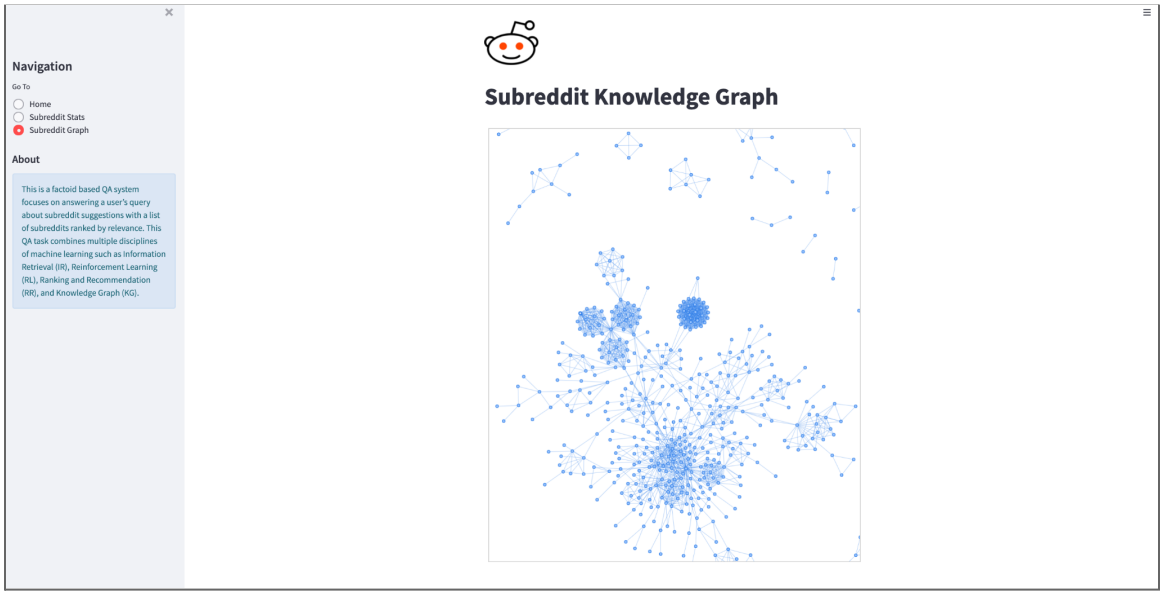

Figure 24: The second (knowledge graph) page shows the organization of reddit communities

The knowledge graph is created from the co-occurring historical questions and answers.

Chapter 6. Testing and Verification

In this project we followed a bottom-up software development approach. Each system component was implemented as a distinct functional unit in a modular, bottom-up fashion. Functional Unit testing was performed on each module and Integration testing for each checkpoint of a system component using an automated testing environment.

6.1 Unit Testing

The different components of the system were unit tested in an automated manner using the PyPl ‘unittest’ module that provides a built-in test framework and test runner to test python development code on multiple unit tests together. Each test failure was recorded in the test log along with necessary details describing the reason for the failure. Critical test failures resulted in testing suspension till the error was resolved. The UnitTesting use cases are presented in Table 4 - Table 8.

Test Case	**Use Case “Post a question”**
Actors	Unit Tester
Goals	Check that the UI provides a functional textbox that accepts a user’s keyed in question
Preconditions	The system is up and running
Actions	Submit a question in the textbox and monitor the request in AWS API Gateway
Postconditions	The system is able to accept the user’s posted question
Open Issues	None

Table 4: Use Case “Post a question”

Test Case	**Read in historically answered questions”**
Actors	Unit Tester
Goals	Check that the UI provides a mechanism for the system developers to upload and query the historically answered questions dataset. Check whether the data collected is valid and reliable.
Preconditions	The system is up and running, including the Reddit APIs and our inference website

Actions	Query samples from the historical answered questions dataset through the UI and cross reference with the official Reddit website.
Postconditions	The system is able to collect, clean, and upload the historically answered questions.
Open Issues	None

Table 5: Use Case "Read in historically answered questions"

Test Case	**"Display reinforcement guided refined questions as suggestions to improve question to get higher-quality answers"**
Actors	Unit Tester
Goals	Check if the UI provides a graphical interface for reinforcement guided question refinement
Preconditions	The system is up and running, historically answered questions dataset has been uploaded and processed, a user question has been posted seeking ranked subreddit recommendations as answers
Actions	Enter question not in the historically answered questions dataset
Postconditions	The system is able to the provide the user with keywords to improve the posted question
Open Issues	None

Table 6. Use Case "Display reinforcement guided refined questions as suggestions to get higher-quality answers"

Test Case	**"Clean user question and historical question(s)"**
Actors	Unit Tester
Goals	Check if the Text Cleaning API is able to clean the text to remove noise and highlight meaningful words, removes empty values and insoluble questions
Preconditions	The system is up and running, historically answered questions dataset has been uploaded and processed, a user question has been posted seeking ranked subreddit recommendations as answers

Actions	Enter empty text or text with common words and punctuations
Postconditions	The data preprocessing stage is able to provide cleaned text for text embedding and vectorization
Open Issues	None

Table 7. Use Case Clean user question and historical question(s)

Test Case	**“Encode user and historically answered questions”**
Actors	Unit Tester
Goals	Check if the user question and historical question(s) are transformed into the appropriate vectorized format with semantic representation and successfully fed to the ML algorithms to learn hidden meanings and draw inferences.
Preconditions	Text Cleaning API produces cleaned and tokenized text data
Actions	Enter text with different lengths to ensure data shape consistency, create a visualization of the semantic representation to ensure similar words are closer to each other
Postconditions	Vectorized user question is used to compute semantic similarity for searching historically similar, answered questions. Vectorized user question are represented in a meaningful way for ML/DL algorithm to draw inference from
Open Issues	None

Table 8. Use Case Encode user and historically answered questions

6.2 Integration Testing

After all functional units passed, integration testing was performed to detect component level bugs. The Integration Testing use case is captured in Table 9.

Test Case	**“Display ranked subreddit recommendations”**
Actors	Integration Test

Goals	Check that the ranked subreddit recommendations as answers and closely related subreddits for added guidance are displayed to the user given a refined question.
Preconditions	The system is up and running, historically answered questions dataset has been uploaded and processed, a user question has been posted seeking ranked subreddit recommendations as answers and reinforcement guided question refinement has been performed.
Actions	Search for a subreddit by entering a description in the textbox
Postconditions	The system is able to display ranked subreddit recommendations
Open Issues	None

Table 9. Display ranked subreddit recommendations

Chapter 7. Performance and Benchmarks

7.1 Normalized Discounted Cumulative Gain

Normalized discounted cumulative gain, also abbreviated as NDCG, is a popular metric to evaluate recommendation and ranking systems. During evaluation, machine learning practitioners often evaluate NDCG after taking the top k recommendations for a given prompt, referred to as NDCG@k. Figure 25. shows the NDCG evaluated for different values of k.

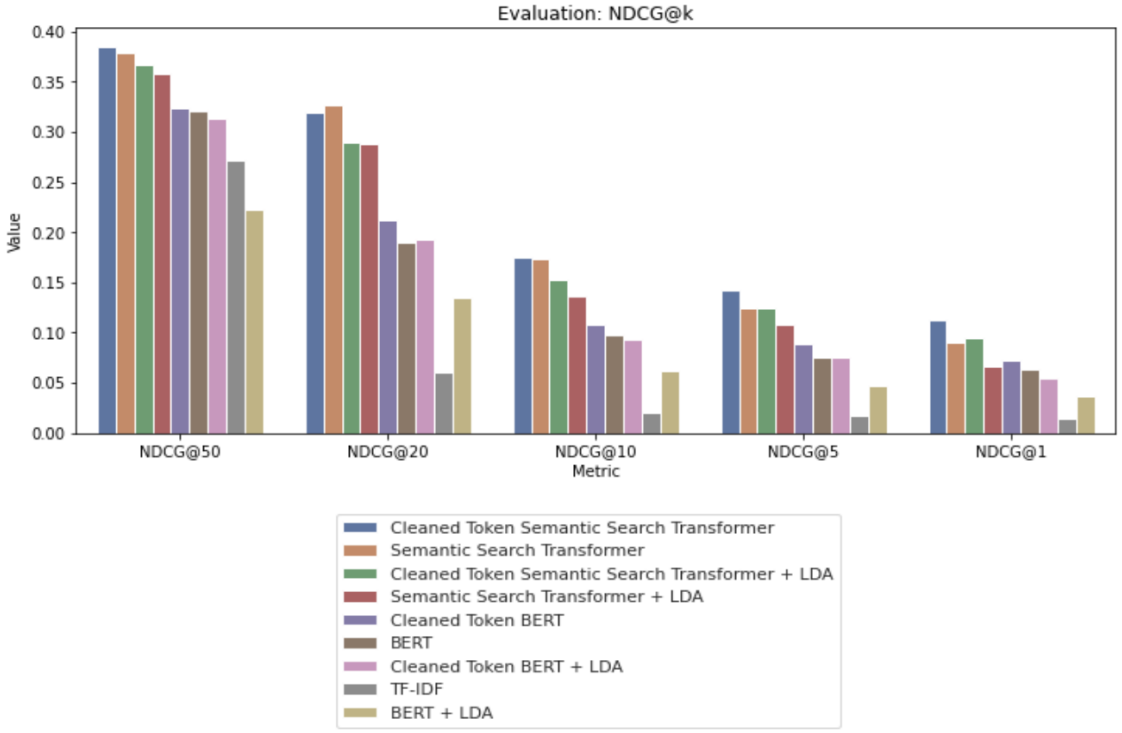

Figure 25. NDCG@k across various experiments

NDCG@k helps evaluate how well the ranking system is recommending the top 3 items versus the top 50 items of a list. NDCG in layman's terms can be thought of as how close the predicted ranking is to the ideal ranking that a user would prefer. Some extra normalization techniques are also applied to this ratio of produced ranking over ideal ranking. Across all our different experiments, our experiment using "Cleaned Token Semantic Search Transformer" performed the best according to the NDCG metric.

7.2 Mean Average Precision

Mean average precision, also abbreviated as MAP, is a popular metric to evaluate recommendation and ranking systems. Figure 26. shows the MAP values evaluated at different values of k.

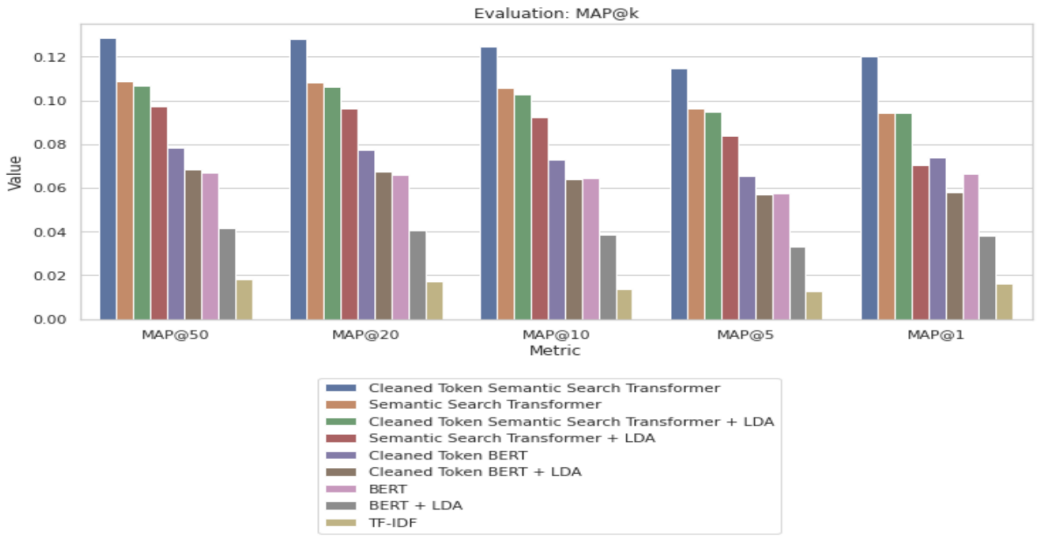

Figure 26. MAP@k across various experiments

During evaluation, machine learning practitioners often evaluate MAP after taking the top K recommendations for a given prompt, referred to as MAP@k. This is in order to evaluate how well the ranking system is recommending the top 3 items versus the top 50 items of a list. MAP is a metric which examines how many items our recommendation system predicted which are actually within the labelled dataset and computes average precision from that. Across all our different experiments, our

experiment using “Cleaned Token Semantic Search Transformer” performed the best according to the MAP metric.

Chapter 8. Deployment, Operations, Maintenance

8.1 CI/CD Pipeline and Cloud deployment

We set up Github workflows in the repository through Github Actions. The workflow is set up to respond to a Github event like a code push to the main branch. Figure 27. shows a snippet of an example github workflow for a successful rollout.

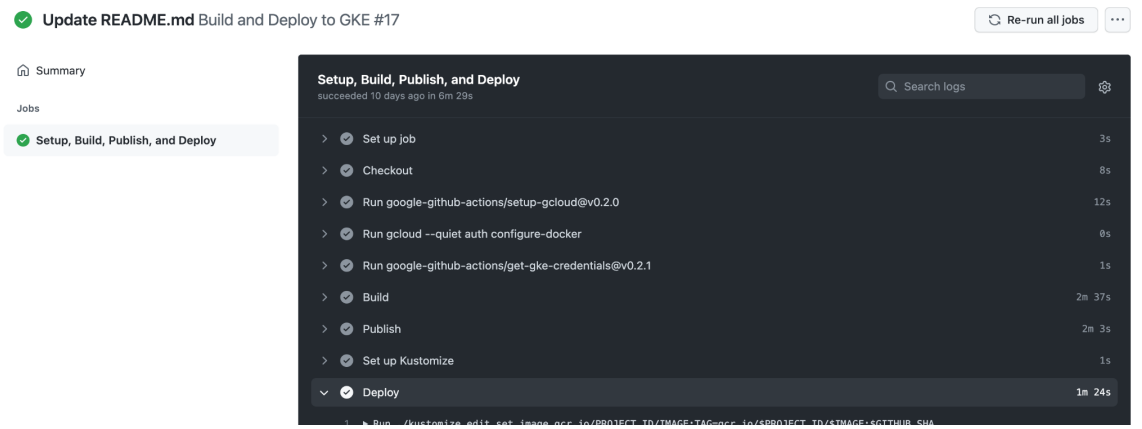

Figure 27. Github actions workflow showing a successful rollout and the IP address at which the application is publicly available

First, a GKE cluster and service account with Identity and Access Management (IAM) roles are set up. Then upon detecting a commit, the Cloud Build takes over the testing and deployment process. It uses the provided ‘Dockerfile’ and runs all the commands in it to build an image of the RQAR system. This image is pushed to the

container registry. The GKE cluster downloads the image from the registry and spins up a node pool to roll out the image. Figure 28. shows the google cloud deployment workflow.

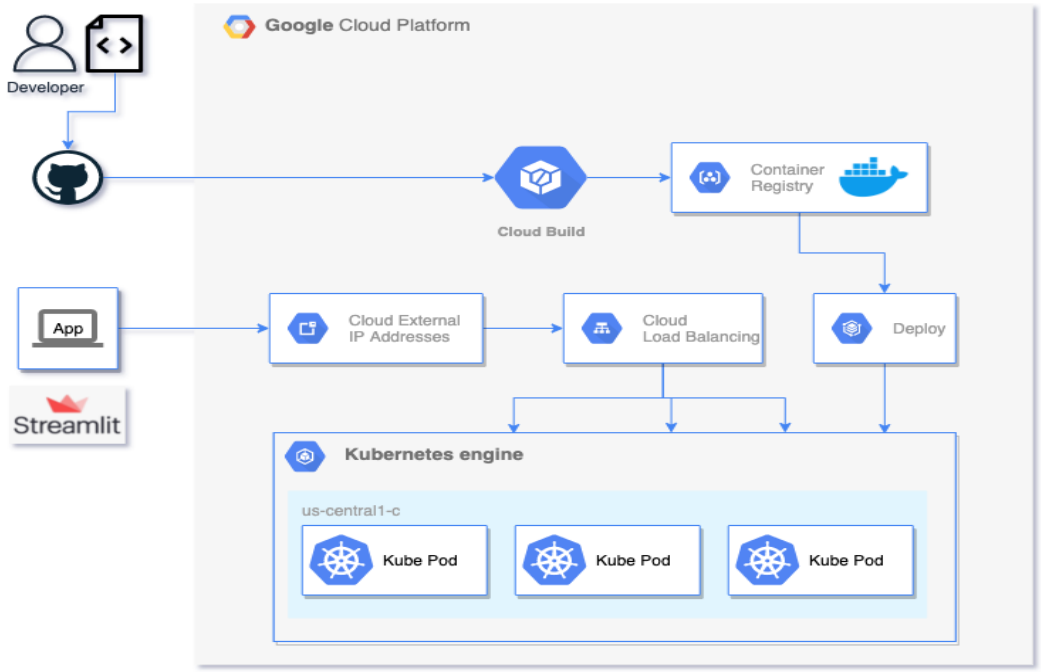

Figure 28. Google Cloud deployment workflow

The Kubernetes settings are provided through 'kustomization.yml', 'deployment.yml ', and 'service.yml' files in the github repository. After the deployment is complete, the web application is accessible through the external IP address provided by the pipeline.

8.2 Operations, Logging and Tracking

Google Cloud platform offers a dashboard to keep track of the Kubernetes clusters and node pools. It also provides a logs dashboard to keep track of the kube pod logs. Figure 29. shows the google cloud dashboard for a kubernetes cluster.

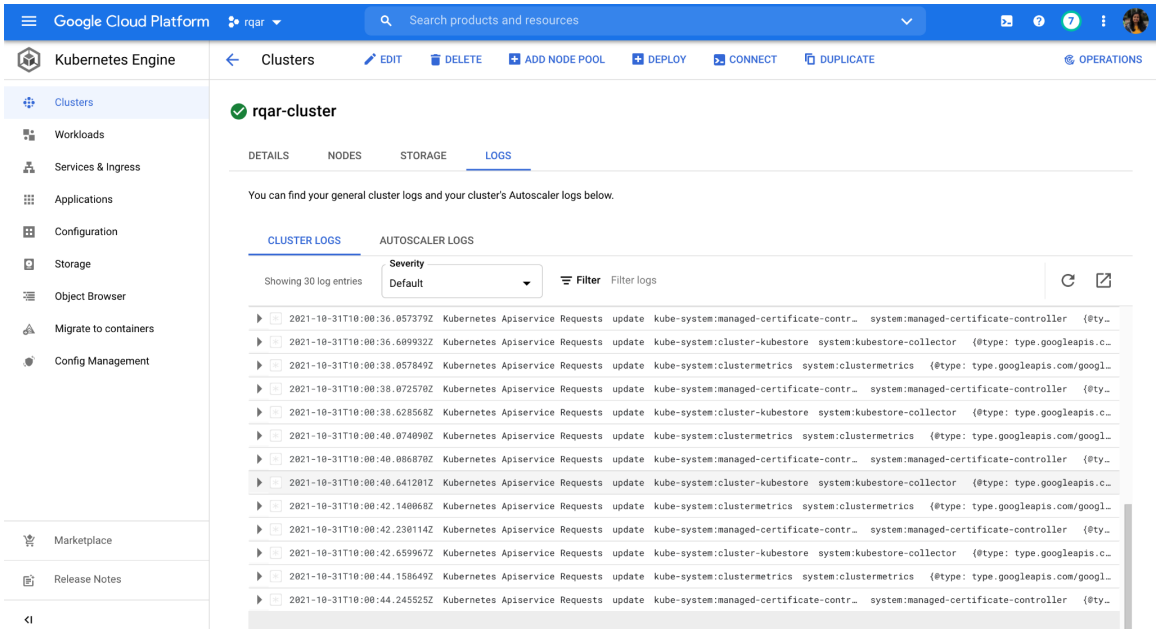

Figure 29. Google Cloud Platform dashboard for a Kubernetes cluster

Chapter 9. Summary, Conclusions, and Recommendations

9.1 Summary

In summary, as more people take to CBQA platforms to find shared interest groups, post questions, crowdsource answers and learn from each other, there is a growing need for a system that responds to community-based network suggestions accurately and efficiently. We found that the existing approaches in this area focused on finding previously answered similar questions or community topic experts to make the

best answer recommendations. While these approaches worked well for platforms when the number of posted questions was few, as the number of questions on CBQA platforms increases, responding to them efficiently is not always possible.

For example on the Reddit platform with over 138,000 active communities, finding the right community to join is challenging. Therefore, we designed and developed a Reinforcement-based Question-Answer Recommender (RQAR) system for responding to community-based network suggestions. Targeted for the Reddit platform, the system uses enhanced contextualization and adaptive reinforcement learning techniques to find and return subreddit recommendations users could post their questions to, to get high-quality responses. In addition, the system uses adaptive learning techniques to provide users guidance to refine their questions for better results.

To prepare a high-quality system, we experimented with many different techniques and compared and contrasted their performance across different models using a “Cleaned Token Semantic Search Transformer” and well-established NDCG and MAP metrics to quantify the quality of a recommendation system.

We additionally productized our model through a web client using Google Cloud Platform and the Kubernetes Engine to serve our predictions to the user on a user friendly website.

9.2 Conclusions

In conclusion, we successfully designed and developed a working machine learning model of a Reinforcement-based Question-Answer Recommender (RQAR) system for responding to community-based network suggestions for the Reddit platform. We evaluated our model and iteratively improved its performance throughout the project. We also created a web application to serve ranked subreddit predictions, provide question refinement suggestions to get high-quality answers, interesting insights about recommended subreddit communities, and an interactive knowledge graph displaying the organization of related subreddit communities in real-time. We deployed our model on the cloud and incorporated latest ML-OP2 functionalities such as automated GIT workflows, Continuous Integration (CI), Continuous Development (CD), Continuous Testing (CT), Model and Pipeline Monitoring, Tracking, and Logging to make the system scalable and easily extendable for the future.

9.3 Recommendations for Further Research

We recommend further research in finding and evaluating more advanced natural language techniques and models for CBQA platforms for both improved Question-Answering and Ranked recommendations.

For improved Question-Answering, some recommendations for further research include using Haystack and Question-Answering bots. For improved ranked recommendations, some recommendations for further research include evaluating

learn-to-rank deep learning techniques now supported by Tensorflow and gradient boosting ranker models.

Glossary

[1] Community-based Question Answering (CBQA) - Question answering system where answers are obtained from the community

[2] Reinforcement-based Question-Answer Recommender System (RQAR) - Question-answer system using various text and natural language context features for signal reinforcement

LIST OF REFERENCES

[1] B. Molina, Reddit is extremely popular. Here's how to watch what your kids are doing, USA Today, July 26, 2018. Accessed on: June 20, 2021. [Online] Available: https://www.usatoday.com/story/tech/talkingtech/2017/08/31/reddit-extremely-popular-heres-how-watch-what-your-kids-doing/607996001

[2] D. V. Vekariya and N. R. Limbasiya, "A Novel Approach for Semantic Similarity Measurement for High Quality Answer Selection in Question Answering using Deep Learning Methods," *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 518-522, doi: 10.1109/ICACCS48705.2020.9074471

[3] R. Tondulkar, M. Dubey and M. S. Desarkar, "Get me the best: predicting best answerers in community question answering sites," *Proceedings of the 2018 ACM International Conference on Recommender Systems (RecSys '16) ser. RecSys '18*, 2018, pp. 251-259, doi: https://doi-org.libaccess.sjlibrary.org/10.1145/3240323.3240346

[4] Patel, S., Reddit Claims 52 Million Daily Users, Revealing a Key Figure for Social-Media Platforms, Wall Street Journal, Dec 1, 2020. Accessed on: July 07,2021. [Online] Available:https://www.wsj.com/articles/reddit-claims-52-million-daily-users-revealing-a-key-figure-for-social-media-platforms-11606822200

[5] Dean, D., Reddit Usage and Growth Statistics: How Many People Use Reddit in 2021?, Backlinko, Feb 25, 2021. Accessed on: July 07, 2021. [Online] Available: https://backlinko.com/reddit-users

[6] A. Arsanjani et al. "Using Spatio-Temporal Similarity to Achieve Deeper Contextualization," *Services Computing Conference 2021*, forthcoming.

[7] A. Arsanjani et al. "Amalgamation: Combining Probabilistic Joins with Similarity Proximity to Increase Performance Metrics for Machine Learning", DeepContext WhitePaper June 2018.

[8] D. Chen et al. “Reading Wikipedia to Answer Open-Domain Questions”, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, doi: https://doi.org/10.18653/v1/p17-1171