

Question Answering over Healthcare Knowledge Graph

Student Name: Anuneet Anand

Roll Number: 2018022

Student Name: Isha Gupta

Roll Number: 2018040

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering
on 13th May, 2022

BTP Track: Engineering

BTP Advisor

Dr.Raghava Mutharaju

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

I hereby declare that the work presented in the report entitled **Question Answering over Healthcare Knowledge Graph** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology in Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr.Raghava Mutharaju**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

.....

Anuneet Anand
Isha Gupta

Place & Date: New Delhi 13th May, 2022

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

.....

Dr.Raghava Mutharaju

Place & Date: New Delhi 13th May, 2022

Abstract

In recent years, there has been a growing interest in digitising health records and healthcare data so that sophisticated systems can be built for faster retrieval, processing and analysis of data. One such type of system is a Question-Answering system which can help the domain experts to retrieve the required data efficiently by asking queries in natural language. Question-answering Systems based on Knowledge Graphs have found applications in various domains. Our work aims to build a question-answering chatbot for electronic health records. The medical professionals can ask questions in natural language, and the Chatbot would iteratively build and execute appropriate SPARQL queries on the Knowledge graph containing the Electronic Health Records to return the relevant data.

Keywords: Question-Answering System, Knowledge Graph, Electronic Health Records, SPARQL Chatbot

Acknowledgments

We want to thank our advisor Dr.Raghava Mutharaju for allowing us to work on this project. His constant support and guidance helped us think critically about the problem and make progress in the project. Undertaking this project enabled us to find our interests in Knowledgeable Computing and Natural Language Processing and enhanced our knowledge about the same.

Work Distribution

We have divided the work equally among ourselves. Both of us collaborated and made meaningful contributions to the project.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Problem Statement	1
2	Literature Review	3
2.1	Quepy	3
2.2	SQG	3
2.3	TR-Discover	4
2.4	NSpM	4
2.5	DBpedia Spotlight	5
3	Experiment	6
3.1	Queries for Evaluation of Annotation	6
3.2	Queries for Evaluation of Translation	6
3.3	Evaluation & Inferences	8
3.3.1	Quepy	8
3.3.2	SQG	8
3.3.3	TR-Discover	8
3.3.4	NSpM	8
3.3.5	DBpedia Spotlight	9
4	Dataset	10
5	Methodology	11
5.1	Initialization	11
5.2	Preprocessing	11
5.3	Entity Recognition	11
5.4	Template Ranking	12
5.5	SPARQL Query Generation	12
5.6	Query Execution	12

6	Architecture	13
6.1	Frontend	13
6.2	Backend	13
7	Evaluation	16
8	Code Availability	21
9	Future Work	22

Chapter 1

Introduction

1.1 Motivation

In recent years, there has been an exponential increase in the amount of data generated across various domains. There is a growing need to develop solutions for managing large scale data and extracting value from it. Knowledge graphs are extensively used to integrate, manage, and generate insights from diverse data sources. A knowledge graph is a graph of data intended to accumulate and convey knowledge of the natural world, whose nodes represent entities of interest and whose edges represent relations between these entities [2]. Several querying languages like SPARQL, Cypher and Gremlin have been developed to efficiently query the knowledge graph and retrieve valuable information. However, most of these querying languages are not intuitive, and only technical professionals can use them to query knowledge graphs.

The benefits of using knowledge graphs to model data can be leveraged in various domains like journalism, law and healthcare. The rapid developments in the biomedical field and the digitization of medical texts and health records have generated vast healthcare data. Modelling this data in knowledge graphs and developing systems for accessing information from it would be an exemplary service to the medical community. Question-answering systems are one such class of systems that can return an answer to a natural language question using the information present in a knowledge graph. Medical professionals can use such a system to retrieve crucial information about drugs, diseases, and patients by providing questions in natural language.

Chatbot-based interacting mechanisms have gained much momentum in the past few years since they are intuitive and accommodate different inputs like text, multi-choice and voice, making them a good choice for question-answering systems.

1.2 Problem Statement

Question-answering systems make extensive use of natural language processing and information retrieval methods. We seek to build a chatbot-like question-answering system that can take a question from the user in natural language, identify important keywords, understand user intent and convert it into an appropriate SPARQL query that can be executed to produce answers to the user.

Several existing question-answering systems are designed to work with generic data as available on DBpedia. Such systems might perform well on simple queries but show poor performance when the user provides domain-specific questions. Hence, we developed a question-answering system focused on healthcare data. While the ontology played a significant role in this, some changes were also required in the grammar and lexicon associated with the system. The question-answering system should provide an intuitive and iterative interaction mechanism for the medical professionals and make it easier for them to retrieve the information appropriately.

In the bigger picture, our solution should be able to work with various healthcare datasets modelled as knowledge graphs. We kept the system modular so that it could be easily integrated into other larger platforms associated with healthcare.

Chapter 2

Literature Review

We conducted an extensive review of knowledge graph-based question-answering systems covering various approaches.

2.1 Quepy

Introduction: Quepy is a python framework developed by Machinalis that can convert natural language questions to SPARQL queries.

Methodology: Quepy uses the nltk tagger and wordnet to process the natural language questions and identify the entities and relations present. It then uses the REFO (Regular Expressions For Object) module to map the entities and relations to SPARQL query templates and generate SPARQL queries. The ontology can be specified in the form of python classes in the domain-specific language file and custom SPARQL templates in the parsing file to use Quepy for a particular domain.

2.2 SQG

Introduction: SQG (SPARQL Query Generator) is a python framework that can build SPARQL queries from natural language questions [7]. It models the question as a walk (sequence of edges) in the knowledge graph, which only encompasses the relevant entities and relations along with potential answers to the question.

Methodology: SQG assumes that entities and relations have already been identified from the natural language question and focuses primarily on SPARQL query building. The first step is to classify the question type like boolean, list or count using Naive Bayes and SVM models. Once the question type is established, a sub-graph consisting of only relevant entities and relations is extracted from the knowledge graph, and all possible walks in this sub-graph are enumerated. Tree-LSTM is used to rank the enumerated walks to return the most suitable one. The ranking is based on the similarity between the syntactic structure of questions (modelled as a dependency parse tree) and the tree structure of the enumerated walks. The best walk is finally used to build the SPARQL query for the given natural language question and generate a suitable answer.

MSQG is a modification of SQG that uses a different neural architecture for calculating the similarity between questions, and the enumerated walks [1]. Instead of using the same Tree LSTM to generate both question and query representations, MSQG uses BERT to generate question representations and Tree LSTM to generate query representations, resulting in slight improvements in the ranking system.

2.3 TR-Discover

Introduction: TR-Discover is a natural language interface for querying interlinked datasets developed by the Research and Development centre of Thomson Reuters [4]. It aims to provide non-technical professionals with an easy and efficient way to query and analyse linked data without the need of learning query languages like SPARQL.

Methodology: The TR-Discover system receives questions from the users in natural language and parses them using a feature-based context-free grammar (FCFG). The non-terminal nodes in the FCFG have phrase structure rules (based on the ontology), and the leaf nodes in the FCFG have lexical entries (generated from the entities in the knowledge graph). The natural language question is converted into a First Order Logic (FOL) representation. A FOL parser is then used to generate a parse tree from the FOL representation and the grammar. The SPARQL query is created by performing an in-order traversal of the generated parse tree. This query can then be executed to return relevant results.

TR-Discover also provides an auto-suggest mechanism to help users construct well structured natural language questions. The auto-suggest component does not use query logs and instead generate suggestions based on the entities and relations present in the knowledge graph. It is built based on the idea of left-corner parsing and rank suggestions using popularity scores of nodes from the RDF graph.

2.4 NSpM

Introduction: NSpM (Neural SPARQL Machine) is a Machine Translation approach for question answering over knowledge graphs [5]. It treats SPARQL as another language and uses a Seq2Seq model to translate natural language questions into sequences that encode appropriate SPARQL queries.

Methodology: The NSpM architecture consists of three main components - Generator, Learner and Interpreter. The model requires a manually annotated dataset of natural language questions and their respective SPARQL queries to create query templates. The generator uses the query templates and RDF triples of a knowledge graph to generate the training dataset for the Learner. Post-training, the Learner component is responsible for converting any natural language question to a sequence that can encode the corresponding SPARQL query. The Interpreter component encodes the sequence received from the Learner component into the SPARQL query. This query can then be executed to return relevant results.

2.5 DBpedia Spotlight

Introduction: DBpedia Spotlight is a tool for automatically annotating mentions of DBpedia resources in text documents with DBpedia URIs [3]. The annotations can be configured to specific needs with the help of DBpedia Ontology and tuning parameters. The user can also specify error tolerance for annotations.

Methodology: When the user provides a text document, it is first parsed to spot phrases that might be a DBpedia resource. DBpedia uses the LingPipe POS tagger and uses a set of DBpedia labels as the lexicon. The candidate resources generated during spotting are ranked using the cosine similarity scores between their context vectors and the surrounding context.

DBpedia offers the following configuration parameters:-

1. **Resource Set to Annotate:** To restrict type of resources to be annotated.
2. **Resource Prominence:** Avoid annotation of rare resources.
3. **Topic Pertinence:** To give importance to topic relevance of annotated resources.
4. **Contextual Ambiguity:** To avoid annotating resources with high ambiguity.
5. **Disambiguation Confidence:** Specifying high Disambiguation Confidence would lower contextual ambiguity threshold.

Chapter 3

Experiment

3.1 Queries for Evaluation of Annotation

We chose the following five simple queries from Task 2 of QALD-4 [6] to evaluate the annotation tools. The queries were manually annotated to highlight the entities present and this was used as a gold standard for evaluation of the tools.

Natural Language Query	Annotated Query
Which genes are associated with breast cancer?	Which genes are associated with breast cancer ?
Which are possible drugs against rickets?	Which are possible drugs against rickets ?
What are side effects of drugs used for asthma?	What are side effects of drugs used for asthma ?
Which drugs have fever as a side effect?	Which drugs have fever as a side effect ?
Give me all diseases of the connective tissue class.	Give me all diseases of the connective tissue class.

3.2 Queries for Evaluation of Translation

We chose the following Natural Language queries designed for our dataset along with their corresponding SPARQL queries to evaluate the translation tools.

Natural Language Query	SPARQL Query
Produce all the Male Patients with chest pain and the date of onset of the symptom of such patients.	<pre> select ?person ?gender ?date_of_onset ?symptom where { ?person a schema:Patient. ?person schema:hasCondition ?indCond. ?person schema:gender "M". ?person schema:gender ?gender. ?indCond a schema:IndividualCondition; schema:conditionType ?condition. ?indCond schema:dateOfOnset ?date_of_onset. ?condition rdfs:label "Chest pain". ?condition rdfs:label ?symptom. } </pre>
What are the distinct drugs administered to people with Dyspnea?	<pre> select ?patient ?drugname where{ ?patient a schema:Patient. ?patient schema:hasCondition ?indCon. ?indCon a schema:IndividualCondition; schema:conditionType ?condition. ?condition rdfs:label "Dyspnea". ?patient schema:drug_administered ?indDrug. ?indDrug schema:drugType ?drug. ?drug rdfs:label ?drugname. } </pre>
What is the date of onset of patients with condition snomed id 433736?	<pre> select (str(?patientId) as ?PatientId) (str(?dateOfOnset) as ?DateOfOnset) ? condition_name where{ ?patient a schema:Patient; schema:hasPatientID ?patientId; schema:hasCondition ?indCon. ?indCon a schema:IndividualCondition; schema:conditionType ?condition. ?indCon schema:dateOfOnset ?dateOfOnset. ?condition ?snomed_id ?snomedID. ?condition rdfs:label ?condition_name. filter(?snomedID="433736"^^xsd:int). } limit 20 </pre>
What are the routes of drug administration for people with age greater than 60 having fever?	<pre> select distinct (str(?method) as ?Method) where{ ?patient a schema:Patient; schema:age ?age. ?patient schema:drug_administered ?indDrug; schema:hasCondition ?indCon. ?indCon schema:conditionType ?condition. ?condition rdfs:label "Fever". ?indDrug schema:routeOfAdministration ?method. Filter (?age>60). } </pre>
Produce all the Female Patients with Carpal tunnel syndrome having age < 50 along with the date of offset of the symptom of such patients.	<pre> select ?person ?gender ?age ?date_of_offset ?symptom where { ?person a schema:Patient; schema:hasCondition ?indCond; schema:gender "F"; schema:gender ?gender; schema:age ?age. ?indCond a schema:IndividualCondition; schema:conditionType ?condition; schema:dateOfOnset ?date_of_offset. ?condition rdfs:label "Carpal tunnel syndrome"; rdfs:label ?symptom. filter (?age<50). } </pre>

Query Prefix	PREFIX ex: < http://www.example.org/btp_ontology/individual/ > PREFIX schema: < http://www.semanticweb.org/btp/clinical/ontology/ > PREFIX xsd: < http://www.w3.org/2001/XMLSchema# > PREFIX rdfs: < http://www.w3.org/2000/01/rdf-schema# >
--------------	---

3.3 Evaluation & Inferences

The given tools were made for generic data, had their code-base not appropriately maintained and posed other hurdles for a uniform evaluation. We tried to evaluate the tools on our set of queries wherever possible and provided a subjective evaluation otherwise. We have also mentioned our inferences from the methodology used by the tool.

3.3.1 Quepy

The project source code is publicly available on GitHub, but it has not been maintained for the past few years and is not supported on Python 3. The tool heavily depends on predefined SPARQL templates. However, manually converting ontology to python classes in the domain-specific language file and designing an exhaustive list of templates is not feasible for large datasets like ours. On evaluating the basic version of it, we observed that it could generate SPARQL queries for some generic natural language queries like "*What is Capital of India?*". However, it failed on all of our biomedical queries. Nonetheless, the project serves as a basic framework that can be used as a reference while building our custom Question Answering system.

3.3.2 SQG

The source code of SQG is publicly available on GitHub, but the repository has not been updated recently. The available implementation had some dependency issues, so we could not evaluate our queries. The implementation also assumes that we have already identified the entities and relations in our question so can not be used directly for our use case. SQG claims to have achieved a macro F1-measure of 75% on the LC-QuAD dataset. Nonetheless, the high-level idea of using syntactic representations of question can be borrowed.

3.3.3 TR-Discover

The source code for TR-Discover is not publicly available and can't be evaluated on our data. However, a naive and undocumented implementation based on the paper is available on GitHub, which uses First-Order Logic for converting natural language questions to SQL queries. TR-Discover was evaluated on Task 2 of QALD-4, which focuses on Biomedical question answering. It achieved an F1-Score of 85%, which is comparable to other state of the art systems. However, TR-Discover showed poor performance on natural language questions which consisted of quantifiers. The auto-suggest component proposed by TR-Discover is a suitable interaction mechanism that can help users build precise queries.

3.3.4 NSpM

The source code of NSpM is publicly available on GitHub, and the contributors actively maintain the repository. However, a large manually annotated dataset of natural language questions and corresponding queries and URIs/RDF triples need to be provided for a domain-specific knowledge graph, which is not feasible. Moreover, the implementation of NSpM is quite complex and cannot be customised easily. Due to the same reason, it failed to show any appreciable performance on our sample set of queries. When NSpM was evaluated on class dbo: Monument of DBpedia, It achieved an accuracy of about 76%. NSpM model has been integrated in Adam Medical Platform at Graphen and in a Telegram Chatbot for DBpedia.

3.3.5 DBpedia Spotlight

The source code of DBpedia spotlight is publicly available on GitHub, and the repository is actively maintained. DBpedia Spotlight also offers RESTful and SOAP web services. Although it is not similar to other tools covered in the literature review, which aim to translate natural language questions to SPARQL queries, DBpedia Spotlight annotation techniques can be leveraged in entity recognition in user questions. We manually annotated our set of queries to highlight the important keywords which can be used in a SPARQL query. We ran these queries on the DBpedia Spotlight tool with difference confidence levels to find out how it highlights the targets. The most relevant keyword marking was done at confidence level 0.3.

Manually Annotated Query/Confidence	0.3	0.5	0.7
Produce all the Male Patients with chest pain and the date of onset of the symptom of such patients .	Produce all the Male Patients with chest pain and the date of onset of the symptom of such patients .	Produce all the Male Patients with chest pain and the date of onset of the symptom of such patients.	Produce all the Male Patients with chest pain and the date of onset of the symptom of such patients.
What are the distinct drugs administered to people with Dyspnea ?	What are the distinct drugs administered to people with Dyspnea ?	What are the distinct drugs administered to people with Dyspnea ?	What are the distinct drugs administered to people with Dyspnea ?
What is the date of onset of patients with condition snomed id 433736 ?	What is the date of onset of patients with condition snomed id 433736?	What is the date of onset of patients with condition snomed id 433736?	What is the date of onset of patients with condition snomed id 433736?
What are the routes of drug administration for people with age greater than 60 having fever ?	What are the routes of drug administration for people with age greater than 60 having fever ?	What are the routes of drug administration for people with age greater than 60 having fever ?	What are the routes of drug administration for people with age greater than 60 having fever ?
Produce all the Female Patients with Carpal tunnel syndrome having age < 50 along with the date of offset of the symptom of such patients .	Produce all the Female Patients with Carpal tunnel syndrome having age < 50 along with the date of offset of the symptom of such patients .	Produce all the Female Patients with Carpal tunnel syndrome having age < 50 along with the date of offset of the symptom of such patients.	Produce all the Female Patients with Carpal tunnel syndrome having age < 50 along with the date of offset of the symptom of such patients.
Which genes are associated with breast cancer ?	Which genes are associated with breast cancer ?	Which genes are associated with breast cancer ?	Which genes are associated with breast cancer ?
Which are possible drugs against rickets ?	Which are possible drugs against rickets ?	Which are possible drugs against rickets ?	Which are possible drugs against rickets ?
What are side effects of drugs used for asthma ?	What are side effects of drugs used for asthma ?	What are side effects of drugs used for asthma ?	What are side effects of drugs used for asthma ?
Which drugs have fever as a side effect ?	Which drugs have fever as a side effect ?	Which drugs have fever as a side effect?	Which drugs have fever as a side effect?
Give me all diseases of the connective tissue class.	Give me all diseases of the connective tissue class.	Give me all diseases of the connective tissue class.	Give me all diseases of the connective tissue class.

Chapter 4

Dataset

The RDF dataset comprises of three turtle files as follows:-

- **Condition_occurence.ttl:** It contains details about the condition of the patients along with their duration. Each condition has a unique snomed_id. The relationship between the patient, a condition experienced by him/her/them and the date of onset and date of offset of the condition has been modelled using blank nodes.
- **drug_exposure.ttl:** It contains details about drugs/medication taken by patients along with their dosage, duration and route of administration. Each drug also has a unique snomed_id. The relationship between patient, drugs administered to him/her/them and duration of administration has been modelled using blank nodes.
- **person.ttl:** It contains details of the patients like their age and gender. The patients have a unique subject_id.

The above turtle files were loaded in a local instance of GraphDB for visualisation. There are 737,689 unique triples in the dataset. GraphDB also provides a SPARQL endpoint to execute our queries on this data. The ontology file associated with the dataset was explored using Protege. In Fig. 4.1, we can see the various classes present in our dataset.

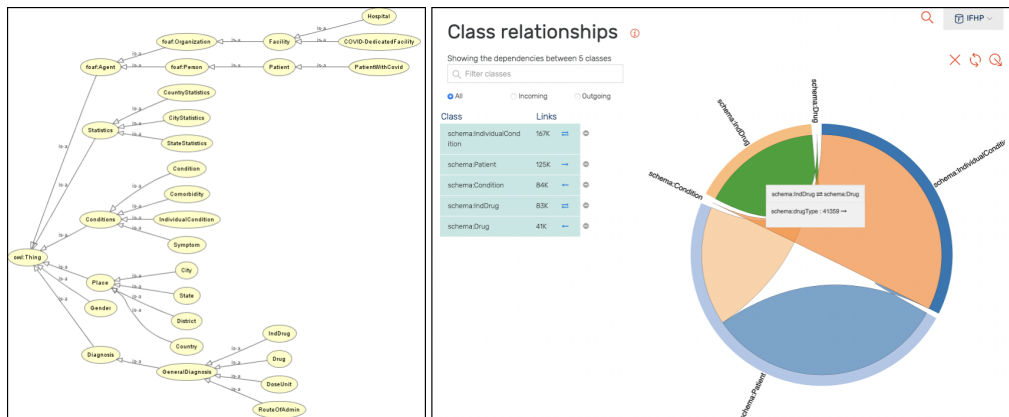


Figure 4.1: Visualising different classes in the Dataset

Chapter 5

Methodology

5.1 Initialization

The user first choose the type of query as specific or generic.

- **Specific queries** are the ones which involve information retrieval about specific entities, like a particular patient or a particular drug.
- **Generic queries** target the entire knowledge base, and filters like age, gender etc., can be added to them.

The user then provides a natural language query that needs to be answered and the type of answer from Record/Count/Boolean. The chatbot stores all this information for further processing.

5.2 Preprocessing

REGEX is used to extract any mention of `snomed_id` or `subject_id` and identify any filters present in the query. The natural language query text is tokenized using the Spacy library. The query tokens are then further processed using the NLTK library, which involves removing the stop-words and lemmatization.

5.3 Entity Recognition

NLTK Wordnet is used for identifying important domain-specific entities in the sentence. For each entity class in our data, we aim to check if there is a token in the query that belongs to that class. We do this by calculating the Wu & Palmer Similarity score between the synset of each class and the synsets of each of the tokens. Wu & Palmer similarity calculates relatedness by considering the depths of the two synsets in the WordNet taxonomies and the LCS depth (Least Common Subsumer). If the class score is over a specified threshold, we confirm the occurrence of an entity of that class in the sentence. The information about the occurrence of different entity classes and corresponding scores are stored for further processing.

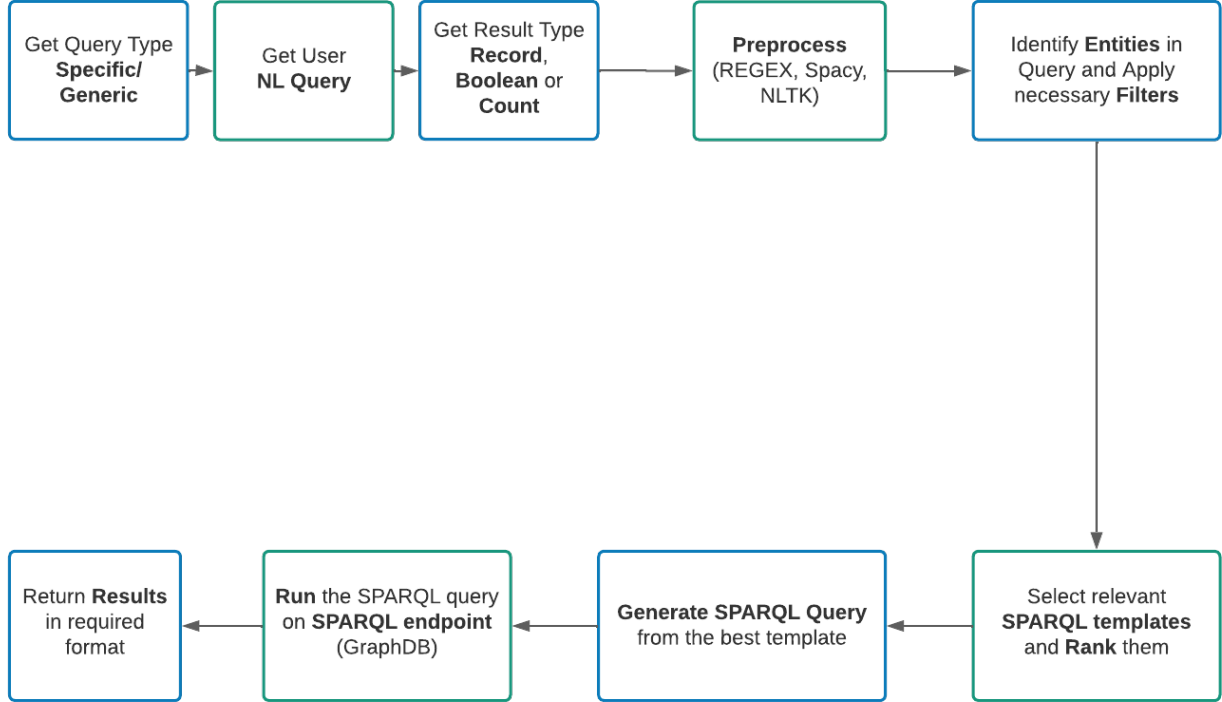


Figure 5.1: Proposed Methodology

5.4 Template Ranking

The information about entity class scores, filters and extracted IDs is used for calculating scores of predefined dataset-dependent SPARQL templates. The score of a template is calculated as the mean score of all entity classes that occur in that template. The templates are then ranked based on these scores and their corresponding descriptions are shown to the user. Among the top templates, the user can choose the template description which is closest to his/her/their natural language query.

5.5 SPARQL Query Generation

The template chosen by the user is populated with the data extracted from the natural language query to generate the required SPARQL query.

5.6 Query Execution

The generated SPARQL query is run on the SPARQL endpoint of a GraphDB instance, which holds our knowledge graphs, and the generated results are returned to the user in a suitable format as requested.

Chapter 6

Architecture

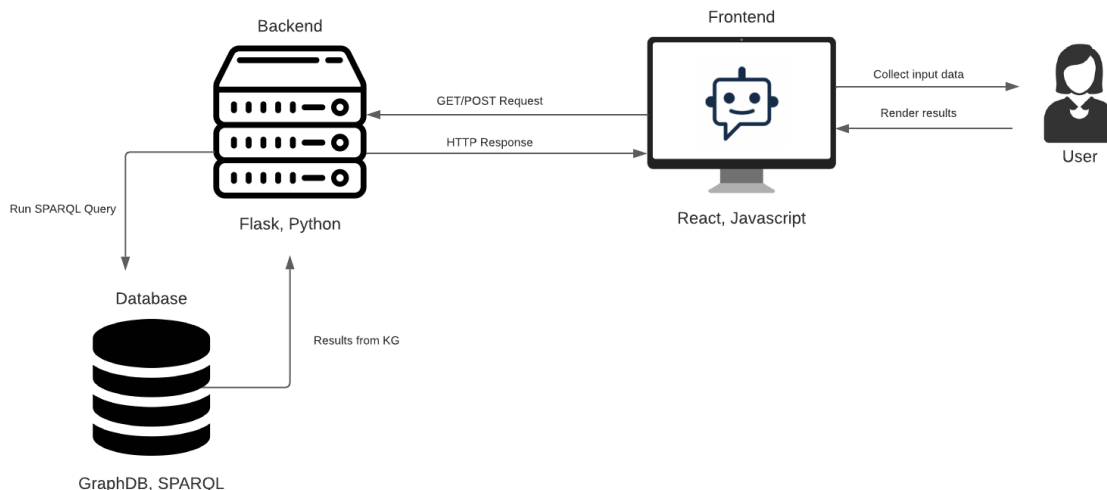
6.1 Frontend

The frontend stack is Reactjs and Javascript + Typescript. We are using a React library called React-Chatbot-Kit for creating our chatbot.

Each message has a handler attached to it, which manages the user's reply. Some replies from the user are text-based, whereas others are inputs via widgets. These responses are collected from the handlers and are sent to the backend to fetch the text messages to be sent from the bot, and the flow finally terminates with the query results being displayed to the user.

6.2 Backend

The backend stack includes a Flask server for handling POST and GET requests and a GraphDB instance for running the SPARQL query over the RDF triples. Natural Language Processing libraries like Spacy ("en_core_web_trf" model) and NLTK are also used at the backend.



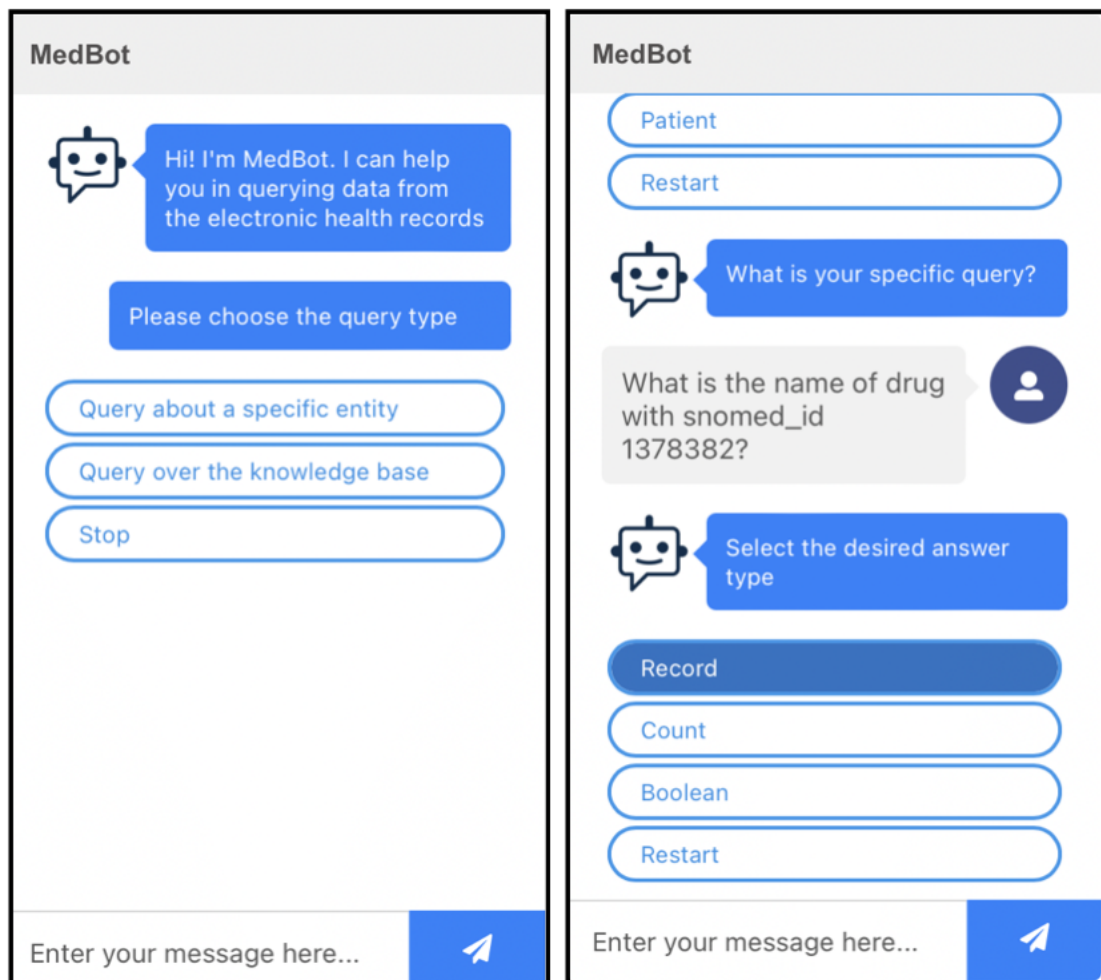



Figure 6.1: Chatbot

MedBot




The following filters were applied automatically :- Age < 18

Enter/Modify the filters for your query

Age

Gender


Continue




Which option which best describes your query?

Filtered Information about patients and drugs given to them

Filtered Information about patients

Enter your message here... 


MedBot



Which option which best describes your query?

Information about drug with Snomed_ID : 1378382


None



Shall I run your query?

Yes

No



Drug Name : Paclitaxel

Please Restart for new query

Restart


Enter your message here... 

Figure 6.2: Chatbot

15

Chapter 7

Evaluation

We prepared a set of 30 natural language queries based on the dataset and used the chatbot to retrieve answers for them. We noted down the SPARQL queries and responses generated by the chatbot and manually evaluated them. We analyse if the chatbot identified all the required entities and did not identify any extra entity. We finally give a binary score to each query based on the relevance of the answer. Please refer to tables on the following pages.

User Query	SPARQL Query	Inference	Score
What is the age of subject_id_100?	SELECT ?age WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_100"^^xsd:str . ?person schema:age ?age . }	The user query was addressed accurately	1
What is the gender of subject_id_4?	SELECT ?age ?gender WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_4"^^xsd:str . ?person schema:age ?age . ?person schema:gender ?gender . }	The required attribute Gender was retrieved but Age was also retrieved.	1
What is the sex of subject_id_2?	SELECT ?age ?gender WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_2"^^xsd:str . ?person schema:age ?age . ?person schema:gender ?gender . }	The required attribute Gender was retrieved but Age was also retrieved.	1
Who is subject_id_1?	NA	No Templates matched	0
I want to know details about a specific patient.	SELECT ?gender WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_111"^^xsd:str . ?person schema:gender ?gender . }	Only Gender was retrieved.	1
Can you tell me age of a specific person?	SELECT ?age ?gender WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_1000"^^xsd:str . ?person schema:age ?age . ?person schema:gender ?gender . }	The required attribute Age was retrieved but Gender was also retrieved.	1
Provide me information about the patient whose ID is subject_id_1.	SELECT ?gender WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_1"^^xsd:str . ?person schema:gender ?gender . }	Only Gender was retrieved.	1
What are the drugs that were given to patient with ID subject_id_2?	SELECT DISTINCT ?drug_name ?dosage_amount ?dosage_unit WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_2"^^xsd:str . ?person schema:drug_administered ?indDrug . ?indDrug schema:drugType ?drugID . ?indDrug schema:patientDosage ?dosage_amount . ?indDrug schema:doseUnit ?dosage_unit . ?drugID rdfs:label ?drug_name . }	The required information was retrieved	1
How many drugs were given to me?	SELECT DISTINCT ?drug_name ?dosage_amount ?dosage_unit WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_1"^^xsd:str . ?person schema:drug_administered ?indDrug . ?indDrug schema:drugType ?drugID . ?indDrug schema:patientDosage ?dosage_amount . ?indDrug schema:doseUnit ?dosage_unit . ?drugID rdfs:label ?drug_name . }	The bot asked to enter Patient ID and retrieved the required information.	1
Provide me information about the drugs that were given to subject_id_2 along with route of administration.	SELECT DISTINCT ?drug_name ?dosage_amount ?dosage_unit ?route ?start ?end WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_2"^^xsd:str . ?person schema:drug_administered ?indDrug . ?indDrug schema:drugType ?drugID . ?indDrug schema:patientDosage ?dosage_amount . }	The required information was retrieved but start and end dates were unnecessary	1

	<pre> ?indDrug schema:doseUnit ?dosage_unit . ?indDrug schema:routeOfAdministration ?route . ?indDrug schema:drug_adm_start ?start . ?indDrug schema:drug_adm_end ?end . ?drugID rdfs:label ?drug_name .] </pre>		
What dose of different drugs were given to subject_id_2?	<pre> SELECT DISTINCT ?drug_name ?dosage_amount ?dosage_unit WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_2"^^xsd:str . ?person schema:drug_administered ?indDrug . ?indDrug schema:drugType ?drugID . ?indDrug schema:patientDosage ?dosage_amount . ?indDrug schema:doseUnit ?dosage_unit . ?drugID rdfs:label ?drug_name . } </pre>	The required information was retrieved	1
What were the start and end dates for different drugs given to subject_id_2?	<pre> SELECT DISTINCT ?drug_name ?dosage_amount ?dosage_unit ?route ?start ?end WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_2"^^xsd:str . ?person schema:drug_administered ?indDrug . ?indDrug schema:drugType ?drugID . ?indDrug schema:patientDosage ?dosage_amount . ?indDrug schema:doseUnit ?dosage_unit . ?indDrug schema:routeOfAdministration ?route . ?indDrug schema:drug_adm_start ?start . ?indDrug schema:drug_adm_end ?end . ?drugID rdfs:label ?drug_name . } </pre>	The required information was retrieved but dosage information was unnecessary	1
For what durations were different drugs given to subject_id_20?	<pre> SELECT DISTINCT ?drug_name ?dosage_amount ?dosage_unit ?start ?end WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_20"^^xsd:str . ?person schema:drug_administered ?indDrug . ?indDrug schema:drugType ?drugID . ?indDrug schema:patientDosage ?dosage_amount . ?indDrug schema:doseUnit ?dosage_unit . ?indDrug schema:drug_adm_start ?start . ?indDrug schema:drug_adm_end ?end . ?drugID rdfs:label ?drug_name . } </pre>	Start and End dates were retrieved but not duration value	0
What were the conditions that subject_id_1 experienced?	<pre> SELECT ?condition_name WHERE { ?person a schema:Patient . ?person schema:hasPatientID "subject_id_1"^^xsd:str . ?person schema:hasCondition ?indCond . ?indCond a schema:IndividualCondition . ?indCond schema:dateOfOnset ?on_set . ?indCond schema:dateOfOffset ?off_set . ?indCond schema:conditionType ?condition . ?condition rdfs:label ?condition_name . } </pre>	The required information was retrieved	1
What is the name of drug with snomed_id 1378382?	<pre> SELECT DISTINCT ?drug_name WHERE { ?drug a schema:Drug . ?drug schema:snomed_id "1378382"^^xsd:integer . ?drug rdfs:label ?drug_name . ?indDrug schema:drugType ?drug . } </pre>	The required information was retrieved	1
Is there a drug with snomed_id 1997881?	<pre> SELECT DISTINCT ?drug_name WHERE { ?drug a schema:Drug . ?drug schema:snomed_id "1997881"^^xsd:integer . ?drug rdfs:label ?drug_name . ?indDrug schema:drugType ?drug . } </pre>	The required information was retrieved	1
Give details about snomed_id 1997881.	<pre> SELECT DISTINCT ?drug_name WHERE { ?drug a schema:Drug . ?drug schema:snomed_id "1997881"^^xsd:integer . ?drug rdfs:label ?drug_name . ?indDrug schema:drugType ?drug . } </pre>	Only drug name was retrieved	1
What is the route of	<pre> SELECT DISTINCT ?drug_name ?route </pre>	The required information was	1

administration of drug with snomed_id 1113073?	<pre>WHERE (?drug a schema:Drug . ?drug schema:snomed_id "1113073"^^xsd:integer . ?drug rdfs:label ?drug_name . ?indDrug schema:drugType ?drug . ?indDrug schema:routeOfAdministration ?route .)</pre>	retrieved	
What is the name of condition with snomed_id 438398?	<pre>SELECT ?condition_name WHERE (?condition a schema:Condition . ?condition schema:snomed_id "438398"^^xsd:integer . ?condition rdfs:label ?condition_name)</pre>	The required information was retrieved	1
Provide information about snomed_id 40.	<pre>SELECT DISTINCT ?drug_name WHERE (?drug a schema:Drug . ?drug schema:snomed_id "40"^^xsd:integer . ?drug rdfs:label ?drug_name . ?indDrug schema:drugType ?drug .)</pre>	The required information was retrieved	1
Tell me the name of conditions experienced by kids under 2.	<pre>SELECT ?patient_id ?age ?gender ?condition_name WHERE (?person a schema:Patient . ?person schema:hasPatientID ?patient_id . ?person schema:age ?age . ?person schema:gender ?gender . FILTER (?age < 2) ?person schema:hasCondition ?indCond . ?indCond a schema:IndividualCondition . ?indCond schema:conditionType ?condition . ?condition rdfs:label ?condition_name .)</pre>	The filter of under 2 was not captured automatically but worked after modifying it.	1
Is there a condition with snomed_id 100?	<pre>SELECT ?condition_name WHERE (?condition a schema:Condition . ?condition schema:snomed_id "100"^^xsd:integer . ?condition rdfs:label ?condition_name)</pre>	The required information was retrieved	1
List information about male patients.	<pre>SELECT ?patient_id ?age ?gender WHERE (?person a schema:Patient . ?person schema:hasPatientID ?patient_id . ?person schema:age ?age . ?person schema:gender ?gender . FILTER (?gender = "M"))</pre>	The required information was retrieved	1
List drugs which have Rectal route of administration.	<pre>SELECT DISTINCT ?patient_id ?age ?gender ?drug_name ?dosage_amount ?dosage_unit WHERE (?person a schema:Patient . ?person schema:age ?age . ?person schema:gender ?gender . ?person schema:drug_administered ?indDrug . ?indDrug schema:drugType ?drugID . ?indDrug schema:patientDosage ?dosage_amount . ?indDrug schema:doseUnit ?dosage_unit . ?indDrug schema:routeOfAdministration "Rectal"^^xsd:str . ?drugID rdfs:label ?drug_name .)</pre>	The required information was retrieved	1
List drugs which were given to kids.	<pre>SELECT DISTINCT ?patient_id ?age ?gender ?drug_name ?dosage_amount ?dosage_unit ?route WHERE (?person a schema:Patient . ?person schema:age ?age . ?person schema:gender ?gender . FILTER (?age < 18) ?person schema:drug_administered ?indDrug . ?indDrug schema:drugType ?drugID . ?indDrug schema:patientDosage ?dosage_amount . ?indDrug schema:doseUnit ?dosage_unit . ?indDrug schema:routeOfAdministration ?route . ?drugID rdfs:label ?drug_name .)</pre>	The required information was retrieved	1
How many drugs have Oral route of administration?	<pre>SELECT DISTINCT ?patient_id ?age ?gender ?drug_name ?dosage_amount ?dosage_unit WHERE (?person a schema:Patient . ?person schema:age ?age . ?person schema:gender ?gender . ?person schema:drug_administered</pre>	The required information was retrieved	1

	<pre> ?indDrug . ?indDrug schema:drugType ?drugID . ?indDrug schema:patientDosage ?dosage_amount . ?indDrug schema:doseUnit ?dosage_unit . ?indDrug schema:routeOfAdministration "Oral"^^xsd:str . ?drugID rdfs:label ?drug_name . } </pre>		
What conditions were experienced by old female patients?	<pre> SELECT ?patient_id ?age ?gender ?condition_name WHERE { ?person a schema:Patient . ?person schema:hasPatientID ?patient_id . ?person schema:age ?age . ?person schema:gender ?gender . FILTER (?age > 60) FILTER (?gender = "F") ?person schema:hasCondition ?indCond . ?indCond a schema:IndividualCondition . ?indCond schema:conditionType ?condition . ?condition rdfs:label ?condition_name . } </pre>	The required information was retrieved	1
What drugs were given to adult male patients?	<pre> SELECT DISTINCT ?patient_id ?age ?gender ?drug_name ?dosage_amount ?dosage_unit ?route WHERE { ?person a schema:Patient . ?person schema:age ?age . ?person schema:gender ?gender . FILTER (?age > 18) FILTER (?gender = "M") ?person schema:drug_administered ?indDrug . ?indDrug schema:drugType ?drugID . ?indDrug schema:patientDosage ?dosage_amount . ?indDrug schema:doseUnit ?dosage_unit . ?indDrug schema:routeOfAdministration ?route . ?drugID rdfs:label ?drug_name . } </pre>	The required information was retrieved	1
How many male patients are there?	<pre> SELECT ?patient_id ?age ?gender WHERE { ?person a schema:Patient . ?person schema:hasPatientID ?patient_id . ?person schema:age ?age . ?person schema:gender ?gender . FILTER (?gender = "M") } </pre>	The required information was retrieved	1
Is there a patient with age > 100?	<pre> SELECT ?patient_id ?age ?gender WHERE { ?person a schema:Patient . ?person schema:hasPatientID ?patient_id . ?person schema:age ?age . ?person schema:gender ?gender . FILTER (?age > 100) } </pre>	The required information was retrieved after manually setting filter	1

Chapter 8

Code Availability

The project is made open source and hosted publicly on GitHub.

Link : <https://github.com/anuneetanand/QAKG-Bot>

The dataset can be provided on reasonable request.

Chapter 9

Future Work

We were able to build a basic working chatbot with an iterative approach towards answering user's healthcare queries. However, this chatbot was restricted to the dataset and ontology we had with us. The chatbot can be extended to work with more datasets and ontology by adding more SPARQL templates. There is also scope of improvement in the entity detection.

The chatbot webapp can be deployed over a server and released for testing among a small userbase, to understand their experience with the chatbot and evaluate how well its answering the user's query and capturing the user's intention.

Bibliography

- [1] DE CARVALHO COELHO, M., DOS REIS, J. C., TÉCNICO-IC-PFG, R., AND DE GRADUAÇÃO, P. F. Learning to build sparql queries from natural language questions.
- [2] HOGAN, A., BLOMQVIST, E., COCHEZ, M., D'AMATO, C., MELO, G. D., GUTIERREZ, C., KIRrane, S., GAYO, J. E. L., NAVIGLI, R., NEUMAIER, S., ET AL. Knowledge graphs. *Synthesis Lectures on Data, Semantics, and Knowledge* 12, 2 (2021), 1–257.
- [3] MENDES, P. N., JAKOB, M., GARCÍA-SILVA, A., AND BIZER, C. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems* (2011), pp. 1–8.
- [4] SONG, D., SCHILDER, F., SMILEY, C., BREW, C., ZIELUND, T., BRETZ, H., MARTIN, R., DALE, C., DUPREY, J., MILLER, T., ET AL. Tr discover: A natural language interface for querying and analyzing interlinked datasets. In *International Semantic Web Conference* (2015), Springer, pp. 21–37.
- [5] SORU, T., MARX, E., MOUSSALLEM, D., PUBLIO, G., VALDESTILHAS, A., ESTEVES, D., AND NETO, C. B. Sparql as a foreign language. *arXiv preprint arXiv:1708.07624* (2017).
- [6] UNGER, C., FORASCU, C., LOPEZ, V., NGOMO, A.-C. N., CABRIO, E., CIMIANO, P., AND WALTER, S. Question answering over linked data (qald-4). In *Working Notes for CLEF 2014 Conference* (2014).
- [7] ZAFAR, H., NAPOLITANO, G., AND LEHMANN, J. Formal query generation for question answering over knowledge bases. In *European semantic web conference* (2018), Springer, pp. 714–728.