A project report on

QUESTION ANSWERING SYSTEM USING KNOWLEDGE GRAPH BASED ON WIKIPEDIA MINED DATA

Natural Language Processing

by

AMRIT GUPTA (17BCE1082)

KRITI GUPTA (17BCE1327)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

May,2020

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

NLTK	Natural Language Toolkit
API	Application Program Interface
RDF	Resource Description Framework

Chapter 1

Introduction

1.1 WIKIPEDIA

Wikipedia is a minefield of information and in most cases the first resource that is referred to when in search of factoid or in answer to any question. It is the first resource that pops up in most searches and therefore the first resource most people click on in order to find answers to their questions.

1.2 KNOWLEDGE GRAPH

A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge. A knowledge graph is a programmatic way to model a knowledge domain with the help of subject-matter experts, data interlinking, and machine learning algorithms. Knowledge graphs could be very useful in question answering based systems as they can relate multiple entities and multiple relations giving rise to many direct, reflective, affirmative and transitive facts and logics.

1.3 QUESTION ANSWERING SYSTEMS

Question answering (QA) is a computer science discipline within the fields of information retrieval and natural language processing (NLP), which is concerned with building systems that automatically answer questions posed by humans in a natural language.

1.4 SYSTEM OVERVIEW

In the case of Wikipedia, more often than not the articles on the subjects are long and convoluted and thus finding what is needed becomes a task in itself. This is the problem we hope to address. We plan to scrape wikipedia documents to create the required corpus and develop a knowledge graph based on it. On this we further plan to create a question answering mechanism that would provide answers to specific questions.

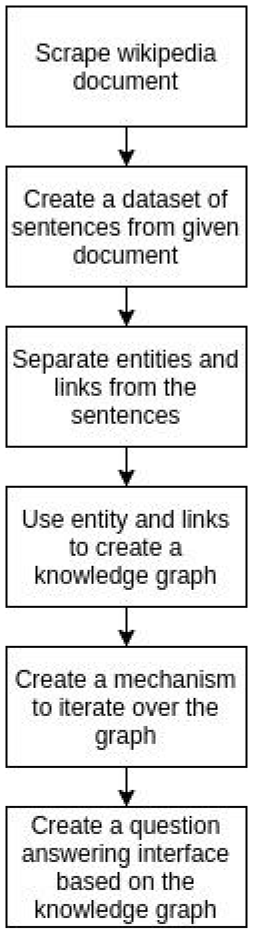

Scrape wikipedia document
Create a dataset of sentences from given document
Separate entities and links from the sentences
Use entity and links to create a knowledge graph
Create a mechanism to iterate over the graph
Create a question answering interface based on the knowledge graph

Figure 1.1 System overview

1.5 PROJECT STATEMENT

Designing and implementing a question answering system using Wikipedia/article based website data. The novelty of our approach is that there is no specific dataset being used to construct the knowledge graph; rather we create a real time knowledge graph based on the user's question. This makes the system highly scalable and versatile as it can answer questions from a wide variety of topics and also don't worry about staying up-to-date.

1.6 OBJECTIVES

1. Find a method to extract data from Wikipedia and other article based sites.
    - Web scrapers
    - Data serving API(s)
2. Data pre-processing and creation of dataset from raw textual data
3. Suitable model to identify the POS tagging
4. Developing rules to extract the actors and relations from the sentences
5. Creating knowledge graph from this dataset
6. Real time scraping/ requesting data based on user question
7. Setting up a pipeline to interface the question and answering system with other modules

1.7 SCOPE

This project could prove to be highly feasible in operational and technical aspects. It could be used to develop an application which provides answers or close answers to user's questions rather than providing them with pages and pages long results. Ofcourse, the present system can only answer basic questions but with future work it can be improved with the help of advanced graph and natural language processing concepts.

Chapter 2

Background

2.1 INTRODUCTION

The extraction of knowledge from different web based resources, such as Wikipedia is an important topic for research as the web is a cesspool of information and navigation is typically a time consuming operation. Question answering systems provide a much better option for users who do not wish to wade through complete documents in search for information. A lot of research has been conducted on these systems in the past and this section aims to collate the insights from different studies regarding question answering systems and extract the gaps which can then be filled in further studies. The first studies on QA systems came up with knowledge based datasets such as WebQuestions and SimpleQuestions. These were very simple in nature and thus higher level queries were handled in multiple ways such as creation of a more complex base, changing the approach to semantic based systems, and knowledge graph embedding. Wikipedia itself was the subject of much research, starting from extraction of lexical knowledge to creation of neural nets for the better parsing of natural language.

2.2 SURVEY ON QUESTION ANSWERING SYSTEMS

2.2.1 WebQuestions And SimpleQuestions

WebQuestions and SimpleQuestions are two datasets created explicitly for the development and improvement of knowledge based question answering systems. These, along with QALD, are some of the most important benchmarks in question answering systems. WebQuestions contains 5810 questions. They can be answered using one reified statement with potentially some constraints like type constraints or temporal constraints. SimpleQuestions contains 108.442 questions which can be answered using one triple pattern. Both of these are used for answering single-relation, factoid questions on finite data. These are fixed datasets and thus cannot be used to resolve questions on newer and ever-changing data which is what is required for our use-case: question answering based on wikipedia documents.

2.2.2 Constraint based Question Answering

Since WebQuestions and SimpleQuestions could not handle multi-constraint questions, a new knowledge based Dataset was devised called ComplexQuestions that could handle the same. It catered to a variety of constraints such as multi-entity constraints, type constraints, temporal constraints, ordinal constraints and aggregation constraints. This new dataset offered better semantic structure and availability of access for multiple connections. A multi-entity query graph could be constructed to denote the knowledge captured. While a step up from WebQuestions and SimpleQuestions, ComplexQuestions is still a fixed database and thus cannot be directly used for our applications. Our data is dynamic and not extremely large. Thus the complexity of overlaying this database over the application is unnecessary.

2.2.3 Semantic Approach to Question Answering

The question answering system in the semantic approach consists of the following 3 main steps:
(1) Triple pattern extraction: Candidate RDF triples are extracted from natural language questions using the dependency graph and POS tags of the questions.
(2) Entity and property extraction: Using RDF triples from the previous step, all subject, predicate and objects are mapped to DBpedia entities, classes or properties.
(3) Answer extraction: Candidate triples are queried over DBPedia and all the answers matching the expected type of a question are ranked and the top result is given as an answer.

2.3 SURVEY ON WIKIPEDIA ANALYSIS

2.3.1 Extracting Lexical Semantic knowledge from the web

This paper brings up the usability and utility of Wikipedia and Wiktionary in the field of NLP. This paper addresses the lack of suitable programmatic access mechanisms to the knowledge stored in these large semantic knowledge bases. They have presented two application programming interfaces for Wikipedia and Wiktionary which are especially designed for mining the rich lexical semantic information dispersed in the knowledge bases, and provide efficient and structured access to the available knowledge. The paper greatly explains about collaborative knowledge bases and its differences and similarities with Linguistic knowledge bases.

2.3.2 Validating answers and using categories in wikipedia effectively

In this paper they have investigated the use of Wikipedia, the open domain encyclopedia, for the Question Answering task. Previous works considered Wikipedia as a resource where to look for the answers to the questions. They have focused on some different aspects of the problem, such as the validation of the answers as returned by their Question Answering System and on the use of Wikipedia "categories" in order to determine a set of patterns that should fit with the expected answer. Validation consists in, given a possible answer, saying whether it is the right one or not. The possibility to exploit the categories of Wikipedia was not considered until then. They performed their experiments using the Spanish version of Wikipedia, with the set of questions of the last CLEF Spanish monolingual exercise. Results showed that Wikipedia is a potentially useful resource for the Question Answering task.

2.3.3 Answering open domain questions using TF-IDF & RCNN from Wiki data

This paper proposes to tackle open domain question answering using Wikipedia as the unique knowledge source: the answer to any factoid question is a text span in a Wikipedia article. This task of machine reading at scale combines the challenges of document retrieval (finding the relevant articles) with that of machine comprehension of text (identifying the answer spans from those articles). The approach combines a search component based on bigram hashing and TF-IDF matching with a multi-layer recurrent neural network model trained to detect answers in Wikipedia paragraphs. They have experimented on multiple existing QA datasets indicate that:

(1) both modules are highly competitive with respect to existing counterparts

(2) multitask learning using distant supervision on their combination is an effective complete system on this challenging task

2.4 SURVEY ON KNOWLEDGE GRAPHS

2.4.1 Knowledge graph Embedding based Question Answering

Question answering over knowledge graph aims to use facts in the knowledge graph to answer natural language questions. It helps end users more efficiently and more easily access the substantial and valuable knowledge in the knowledge graph, without knowing its data structures. Question answering knowledge graphs is a nontrivial problem since capturing the semantic meaning of natural language is difficult for a machine.. Knowledge graph embedding targets at learning a low-dimensional vector representation for each predicate/entity in a knowledge graph, such that the original relations are well preserved in the vectors. These learned vector representations could be employed to complete a variety of downstream applications efficiently. Examples include knowledge graph completion, recommender systems, and relation extraction.

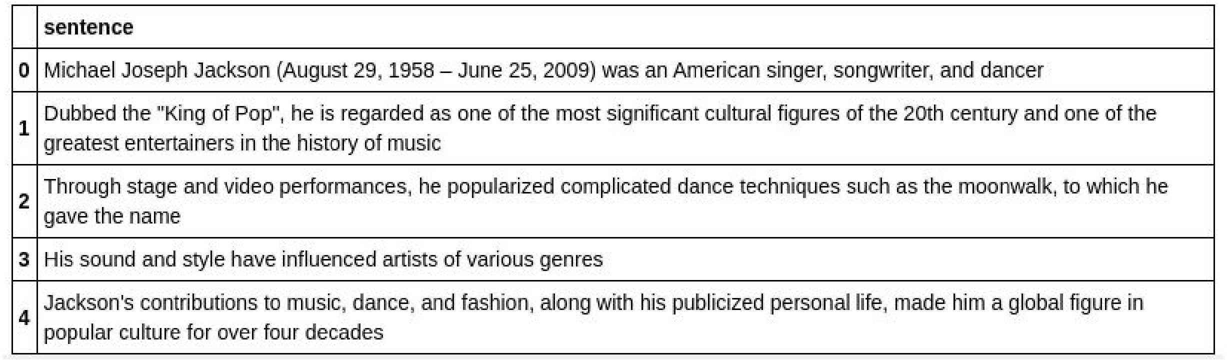
Chapter 3

Algorithms and Design

3.1 DATASET

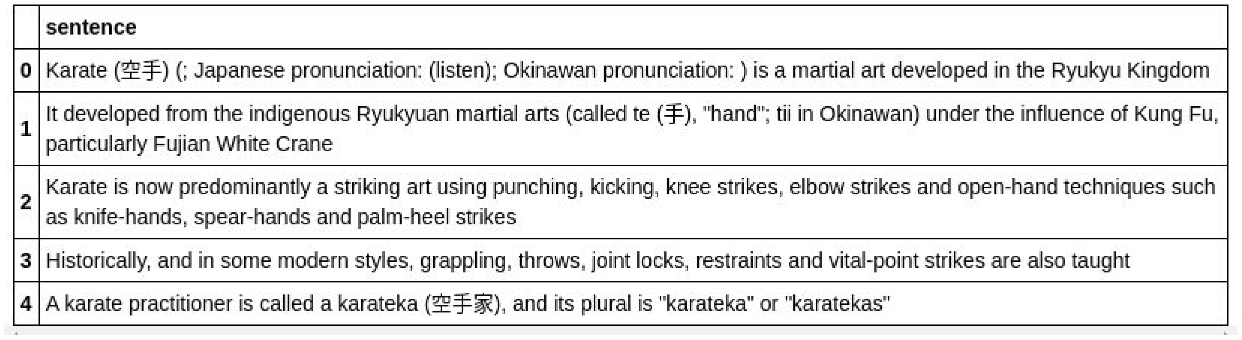
We haven't used a particular dataset for our project, this being the novelty of our system we create a dataset everytime the user sends a query. This enables us to be up-to-date and scalable. So, as the user sends in the question the keywords are identified and related pages are mined on the web using APIs or scrapers and the data is brought in. This data is then used to create our dataset.

Example:



	sentence
0	Michael Joseph Jackson (August 29, 1958 – June 25, 2009) was an American singer, songwriter, and dancer
1	Dubbed the "King of Pop", he is regarded as one of the most significant cultural figures of the 20th century and one of the greatest entertainers in the history of music
2	Through stage and video performances, he popularized complicated dance techniques such as the moonwalk, to which he gave the name
3	His sound and style have influenced artists of various genres
4	Jackson's contributions to music, dance, and fashion, along with his publicized personal life, made him a global figure in popular culture for over four decades

Table 3.1 Samples from real-time "Michael Jackson" dataset



	sentence
0	Karate (空手) (; Japanese pronunciation: (listen); Okinawan pronunciation:) is a martial art developed in the Ryukyu Kingdom
1	It developed from the indigenous Ryukyuan martial arts (called te (手), "hand"; tii in Okinawan) under the influence of Kung Fu, particularly Fujian White Crane
2	Karate is now predominantly a striking art using punching, kicking, knee strikes, elbow strikes and open-hand techniques such as knife-hands, spear-hands and palm-heel strikes
3	Historically, and in some modern styles, grappling, throws, joint locks, restraints and vital-point strikes are also taught
4	A karate practitioner is called a karateka (空手家), and its plural is "karateka" or "karatekas"

Table 3.2 Samples from real-time "Karate" dataset

3.2 METHODOLOGY

3.2.1 Question POS and topic identification

The first step in our pipeline is to receive the question sentence from the user and identify the POS tags in it. After identifying the POS tags we extract the entities and relations. One of these entities are question words such as who, when, which, how etc. We concentrate on the other entities as they carry the relevance for our search.

Example: The user poses the question “Who performed at Wembley?”

We find that the entities are ‘Who’ and ‘Wembley’ and the relation is ‘performed at’. From this we concentrate solely on ‘Wembley’ and the relation ‘performed at’ for the further processing.

3.2.2 Data mining and extraction

After identifying our keywords we move ahead and search for these topics. This is a challenging step, for this problem we have found two solutions. If the domain to be searched provides an API use it else design a manual scraper. The scraper scrapes the site to find relevant pages as close to the keywords and the data is scraped and extracted into the pipeline. The API on the other hand is helpful in getting all the pages with the keywords or synonymous with the keywords. Once we locate the URL of these pages, we request the server for the data.

3.2.3 Data preprocessing

As the received data is a huge collection of paragraphs we need to clean it and organize it. We use the python data processing tools for this. We perform many steps to remove redundant tags, and needless symbols. We split the paragraph into sentences with the help of identifying sentence breaks. Finally we organize the data in a row column format for the next steps.

3.2.4 POS tagging and entity relation identification

The POS tags from each sentence are then identified using NLP models. We then need to introduce our own rules to identify the relevant relations and entities. After these rules are applied and the POS tag patterns are matched, we form a dataset of relations and entities.

3.2.5 Knowledge Graph

After construction of the entity relation dataset we create the knowledge graph. The entities are taken as nodes and the relations as edges. This helps us in identifying many underlying relations which would have been time consuming if manually gone through the whole document.

3.2.6 Answering the user question

In the final step, we use wordnet to find matching synonymous from the verbs specified in the question by the user. We then query the knowledge graph and extract the possible nodes/entities which could be the answer. After checking all possible answers we display the best values to the user.

3.3 SYSTEM ARCHITECTURE

3.3.1 Low Level UML Diagram

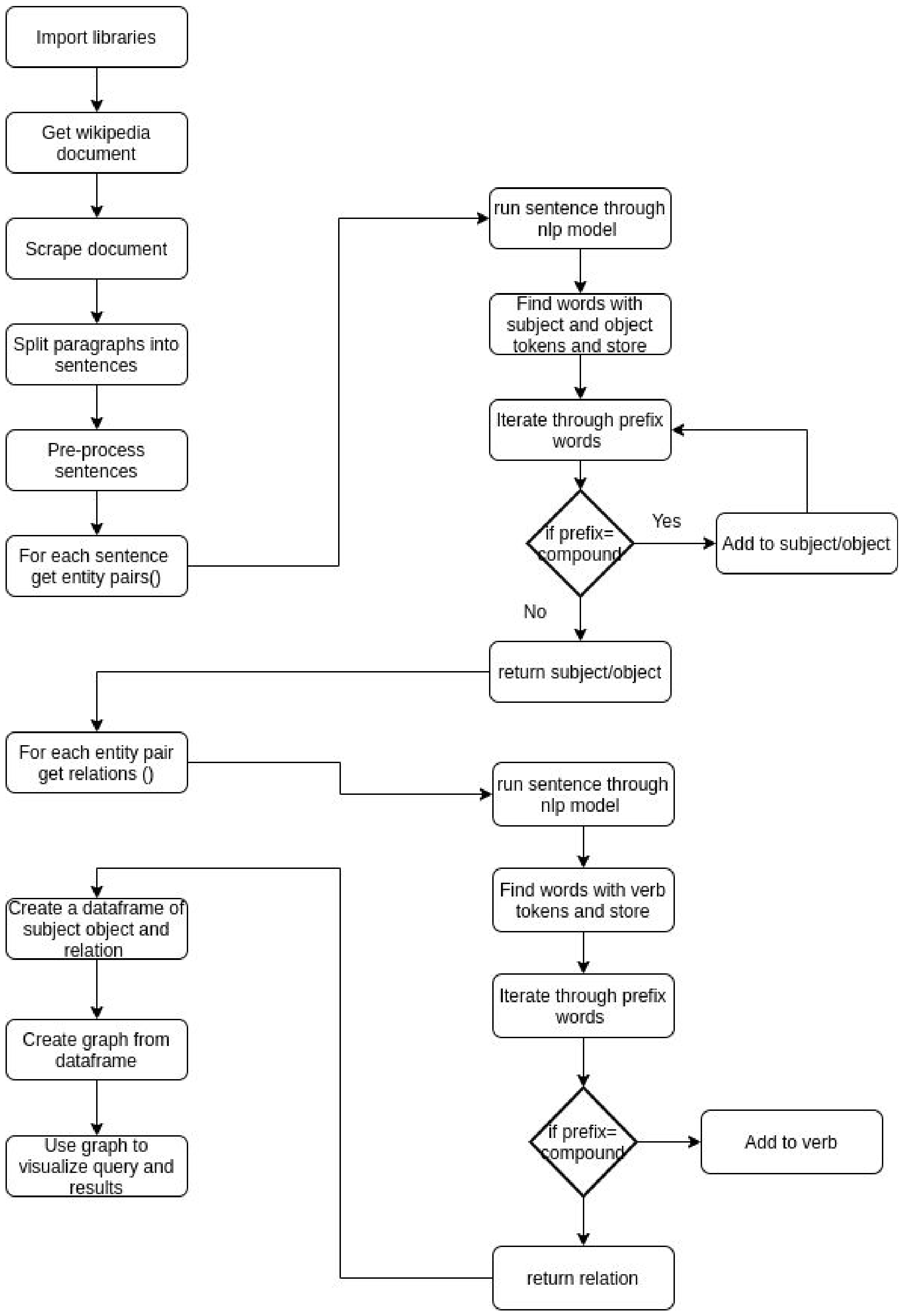

Import libraries
Get wikipedia document
Scrape document
Split paragraphs into sentences
Pre-process sentences
For each sentence get entity pairs()
run sentence through nlp model
Find words with subject and object tokens and store
Iterate through prefix words
if prefix= compound
Yes
Add to subject/object
No
return subject/object
For each entity pair get relations ()
run sentence through nlp model
Find words with verb tokens and store
Iterate through prefix words
if prefix= compound
Add to verb
return relation
Create a dataframe of subject object and relation
Create graph from dataframe
Use graph to visualize query and results

Figure 3.1 Low level UML

3.3.2 High Level UML Diagram

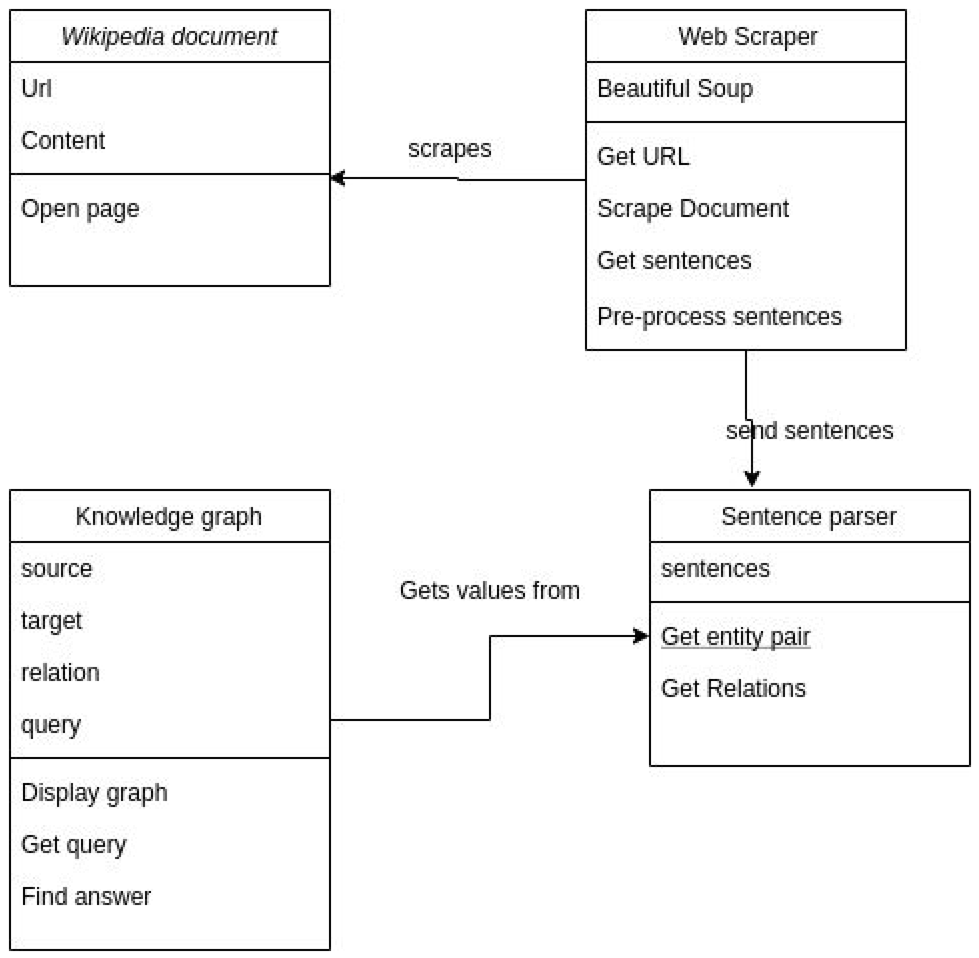


Figure 3.2 High Level UML

3.4 TECHNOLOGY STACK

Language

- Python
- Jupyter notebook (i-python)

Libraries

- Scraping

    - Requests
    - Beautiful Soup
- Data processing
    - Pandas
    - Numpy
- Visualisation
    - Matplotlib
- Knowledge Graph
    - Networkx
- NLP
    - Nltk
    - Spacy

Third-party

- Python-Wikipedia
- Wikipedia api

Models

- ‘en_core_web_sm’ by spacy

Chapter 4

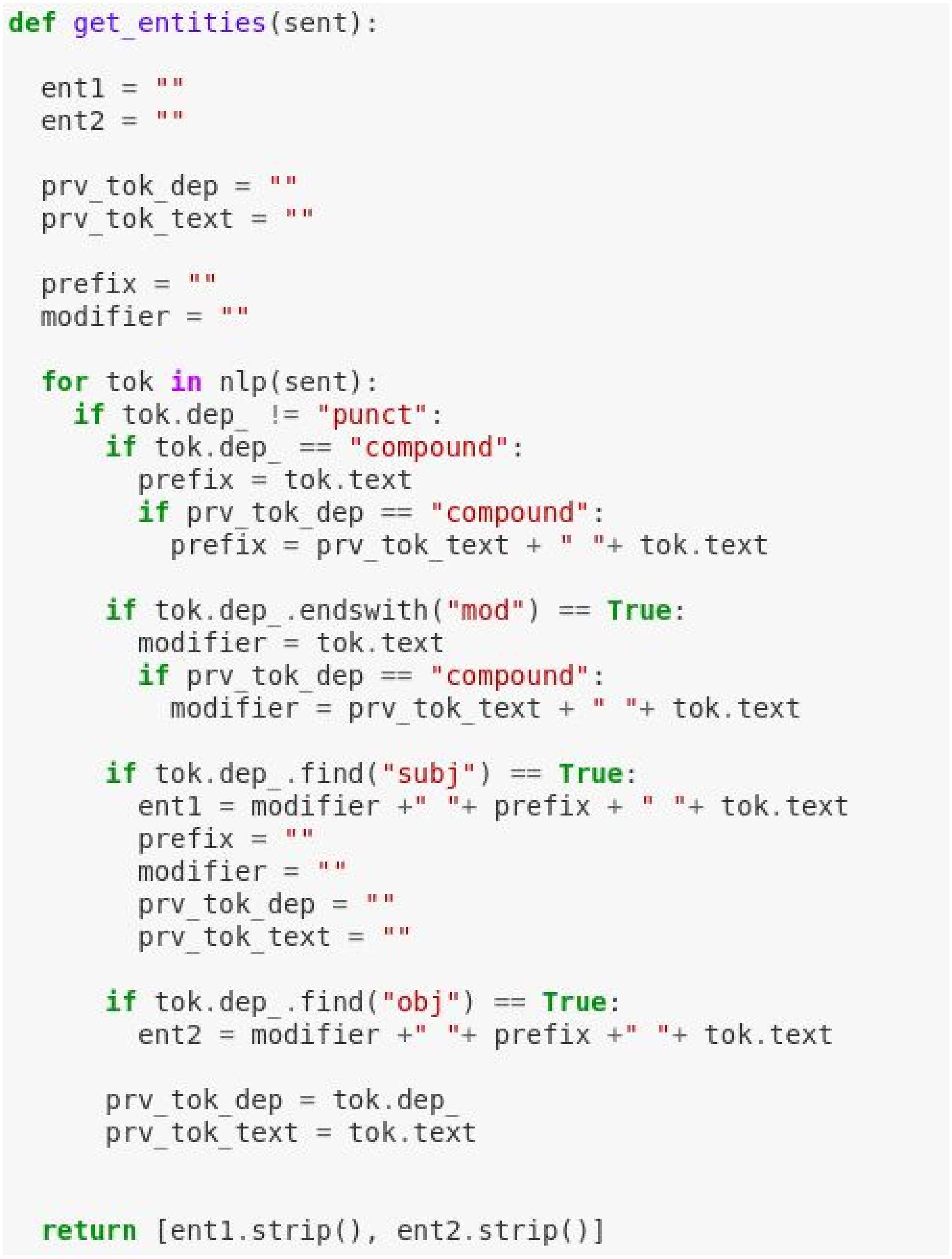
Implementation and Results

4.1 IMPLEMENTATION

The first step towards creating a knowledge graph for a question answering system is the extraction of relevant entities and relations from a sentence. To this end, we created 2 functions to create a list of the subject and object of a sentence as the entities and the verbs as its relations.

For the entities:

The function below first tokenizes the sentence using the nltk library. It then extracts the subject and object in the list of tokens. Since a subject and object can be multiple words (for example, a beautiful red flower is different from just a flower), it iterates through the sentence and adds any compounding words to the entities as a prefix. Since we are operating solely for the English language, all modifiers are prefix words and we do not need to worry about words after the main subject or object.



```python
def get_entities(sent):

  ent1 = ""
  ent2 = ""

  prv_tok_dep = ""
  prv_tok_text = ""

  prefix = ""
  modifier = ""

  for tok in nlp(sent):
    if tok.dep_ != "punct":
      if tok.dep_ == "compound":
        prefix = tok.text
        if prv_tok_dep == "compound":
          prefix = prv_tok_text + " "+ tok.text

      if tok.dep_.endswith("mod") == True:
        modifier = tok.text
        if prv_tok_dep == "compound":
          modifier = prv_tok_text + " "+ tok.text

      if tok.dep_.find("subj") == True:
        ent1 = modifier +" "+ prefix + " "+ tok.text
        prefix = ""
        modifier = ""
        prv_tok_dep = ""
        prv_tok_text = ""

      if tok.dep_.find("obj") == True:
        ent2 = modifier +" "+ prefix +" "+ tok.text

      prv_tok_dep = tok.dep_
      prv_tok_text = tok.text


  return [ent1.strip(), ent2.strip()]
```

For the relations:

A similar process to the entities is used, but this time with the verbs and modifying adverbs.

```python
def get_relation(sent):

  doc = nlp(sent)

  # Matcher class object
  matcher = Matcher(nlp.vocab)

  #define the pattern
  pattern = [{'DEP':'ROOT'},
            {'DEP':'prep','OP':"?"},
            {'DEP':'agent','OP':"?"},
            {'POS':'ADJ','OP':"?"},]

  matcher.add("matching_1", None, pattern)

  matches = matcher(doc)
  k = len(matches) - 1
  if(k>0):
    span = doc[matches[k][1]:matches[k][2]]
    return(span.text)
```

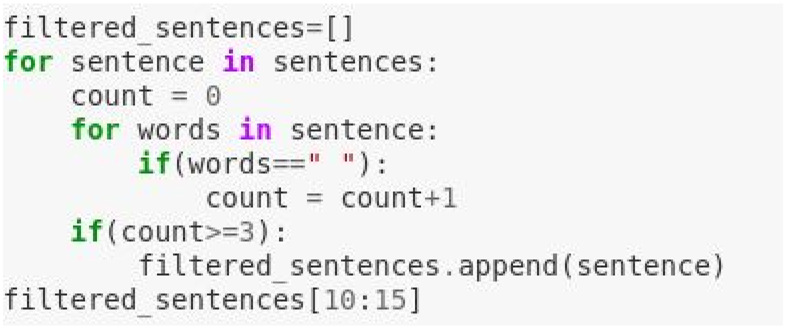
The query is then taken in from the user and processed through these 2 functions. Now since the query will have Wh- words that signify a question, these are removed as they are not needed as an entity.

These entities are then sent through the wikipedia api which returns the web pages related to the words sent.

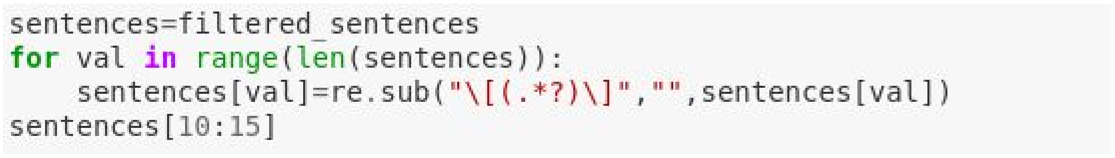
```python
search_results=wikipedia.search(concept, results=3)
wiki_wiki = wikipediaapi.Wikipedia(language='en',extract_format=wikipediaapi.ExtractFormat.WIK
I)
p_wiki = wiki_wiki.page(search_results[0])
text_data=p_wiki.text
```

The results gleaned from the wikipedia pages are in a plaintext format. This needs to be parsed and filtered before sending to the entities and relations functions.

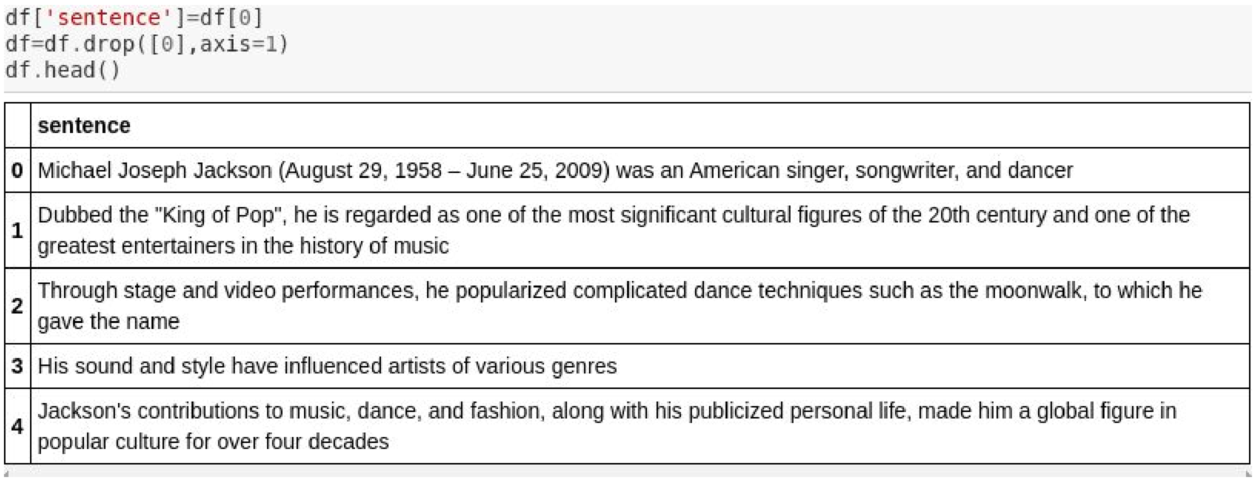
```python
sentences = []
for i in li:
    l = i.split('.')
    for j in l:
        sentences.append(j)
sentences[10:15]
```



```
filtered_sentences=[]
for sentence in sentences:
    count = 0
    for words in sentence:
        if(words==" "):
            count = count+1
    if(count>=3):
        filtered_sentences.append(sentence)
filtered_sentences[10:15]
```



```
sentences=filtered_sentences
for val in range(len(sentences)):
    sentences[val]=re.sub("\[(.*?)\]","",sentences[val])
sentences[10:15]
```



```
df['sentence']=df[0]
df=df.drop([0],axis=1)
df.head()
```

	sentence
0	Michael Joseph Jackson (August 29, 1958 – June 25, 2009) was an American singer, songwriter, and dancer
1	Dubbed the "King of Pop", he is regarded as one of the most significant cultural figures of the 20th century and one of the greatest entertainers in the history of music
2	Through stage and video performances, he popularized complicated dance techniques such as the moonwalk, to which he gave the name
3	His sound and style have influenced artists of various genres
4	Jackson's contributions to music, dance, and fashion, along with his publicized personal life, made him a global figure in popular culture for over four decades

Now, for each sentence in the document, an entity pair and relation is created and stored using the functions defined in the beginning.

```
entity_pairs = []
for i in tqdm(candidate_sentences["sentence"]):
  entity_pairs.append(get_entities(i))
```

```
100%|██████████| 675/675 [00:04<00:00, 140.69it/s]
```

```
entity_pairs[:5]
```

```
[['Michael Joseph Jackson', ''],
 ['he', 'greatest  music'],
 ['such dance he', 'name'],
 ['sound', 'various  genres'],
 ['personal  him', 'four  decades']]
```

```
relations = [get_relation(i) for i in tqdm(candidate_sentences['sentence'])]
```

```
100%|██████████| 675/675 [00:04<00:00, 139.56it/s]
```

A directed graph is created using the entity pairs as nodes and the relations as the edges. The direction of each edge goes from the subject to the object.

```
# extract subject
source = [i[0] for i in entity_pairs]

# extract object
target = [i[1] for i in entity_pairs]

kg_df = pd.DataFrame({'source':source, 'target':target, 'edge':relations})
```

```
# create a directed-graph from a dataframe
G=nx.from_pandas_edgelist(kg_df, "source", "target",
                          edge_attr=True, create_using=nx.MultiDiGraph())
```

```
plt.figure(figsize=(12,12))

pos = nx.spring_layout(G)
nx.draw(G, with_labels=True, node_color='skyblue', edge_cmap=plt.cm.Blues, pos = pos)
plt.show()
```

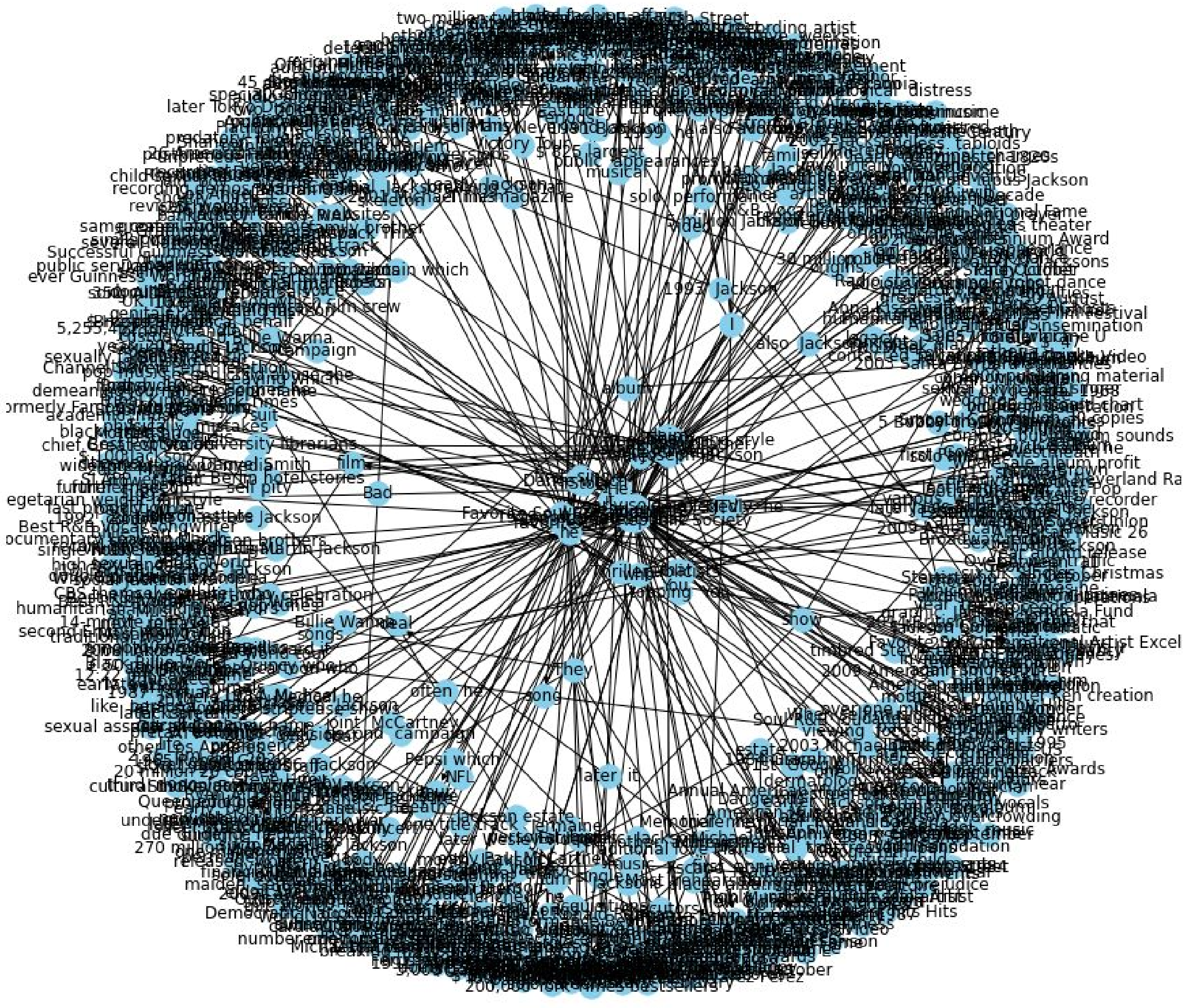
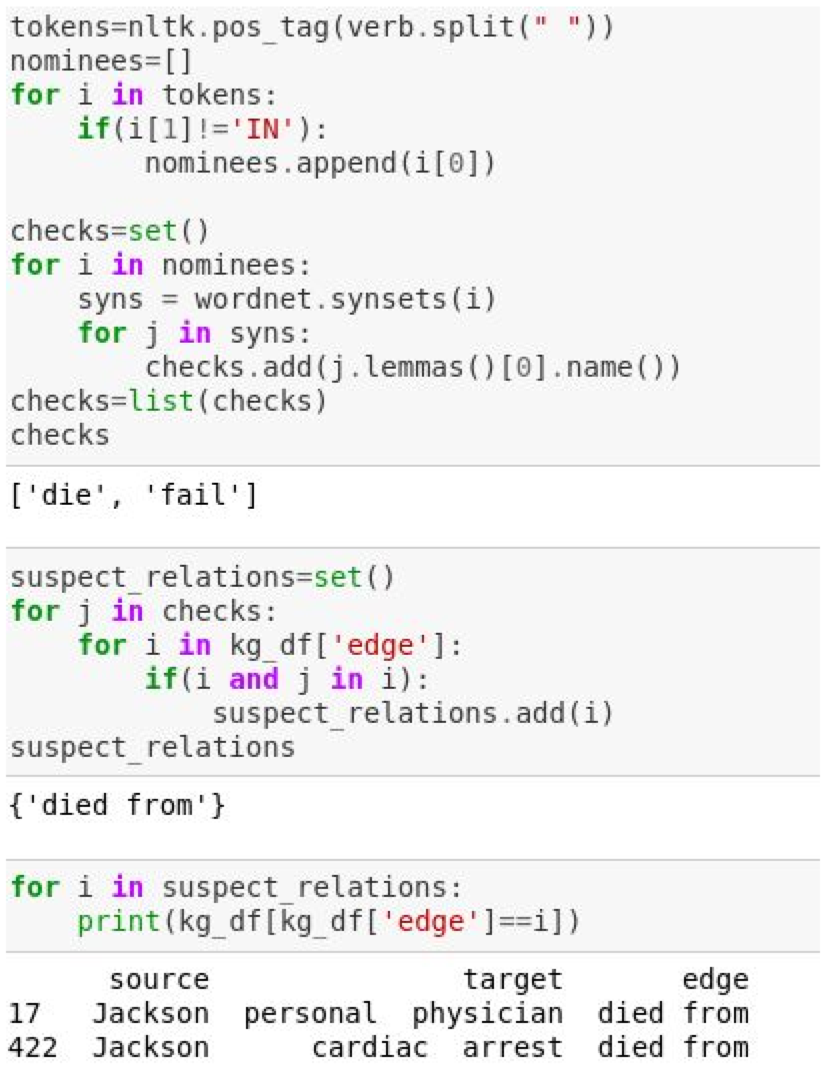


Figure 4.1 Knowledge graph

Then, the relations from the query are taken and passed through WordNet synsets to get similar words and expand the query meaning. The gleaned words are checked against every relation present in the graph and if they match or are similar to any of them, the source, target and relation are printed



```
tokens=nltk.pos_tag(verb.split(" "))
nominees=[]
for i in tokens:
    if(i[1]!='IN'):
        nominees.append(i[0])

checks=set()
for i in nominees:
    syns = wordnet.synsets(i)
    for j in syns:
        checks.add(j.lemmas()[0].name())
checks=list(checks)
checks
```

```
['die', 'fail']
```

```
suspect_relations=set()
for j in checks:
    for i in kg_df['edge']:
        if(i and j in i):
            suspect_relations.add(i)
suspect_relations
```

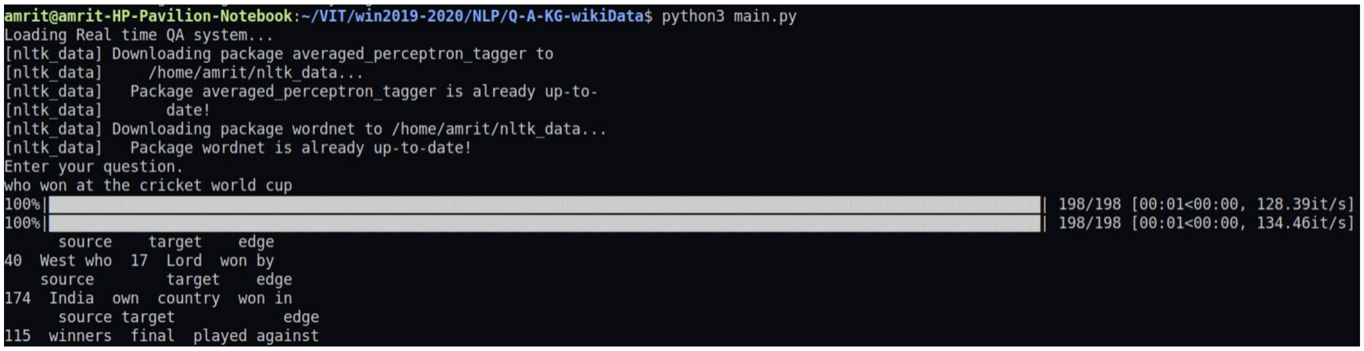
```
{'died from'}
```

```
for i in suspect_relations:
    print(kg_df[kg_df['edge']==i])
```

```
      source                  target       edge
17   Jackson  personal  physician  died from
422  Jackson       cardiac    arrest  died from
```

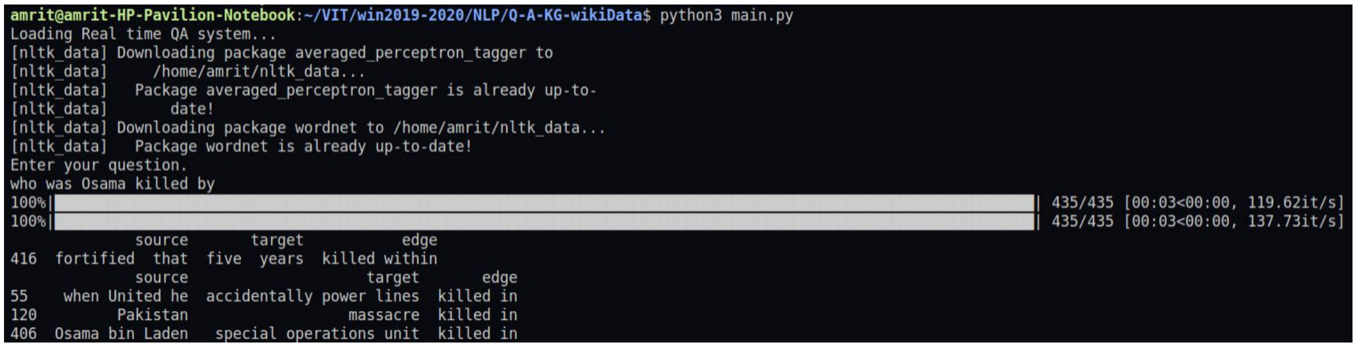
Thus, the knowledge in a document is parsed down to 5-6 relevant words, which are close to the answer of the user's question.

We have consolidated the jupyter code into a python application to make it end-to-end and more user friendly. The following are a few test cases the system could answer if not was very close.



```
amrit@amrit-HP-Pavilion-Notebook:~/VIT/win2019-2020/NLP/Q-A-KG-wikiData$ python3 main.py
Loading Real time QA system...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/amrit/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /home/amrit/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Enter your question.
who won at the cricket world cup
100%|████████████████████████████████████████████████████████| 198/198 [00:01<00:00, 128.39it/s]
100%|████████████████████████████████████████████████████████| 198/198 [00:01<00:00, 134.46it/s]
      source    target    edge
40  West who  17  Lord  won by
    source        target    edge
174  India  own  country  won in
      source target             edge
115  winners  final  played against
```

Figure 4.2 Output for query "Who won at the World Cup?"



```
amrit@amrit-HP-Pavilion-Notebook:~/VIT/win2019-2020/NLP/Q-A-KG-wikiData$ python3 main.py
Loading Real time QA system...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/amrit/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /home/amrit/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Enter your question.
who was Osama killed by
100%|████████████████████████████████████████████████████████| 435/435 [00:03<00:00, 119.62it/s]
100%|████████████████████████████████████████████████████████| 435/435 [00:03<00:00, 137.73it/s]
            source       target            edge
416  fortified  that  five  years  killed within
            source                    target       edge
55    when United he  accidentally power lines  killed in
120         Pakistan                  massacre  killed in
406  Osama bin Laden   special operations unit  killed in
```

Figure 4.3 Output for query "Who was Osama killed by?"

The full source code along with the notebooks and outputs is available in the GitHub repository: https://github.com/crescent-igor/QA-KGraph-wikiData

4.2 RESULTS

A end to end system has been created that takes in a query, scrapes Wikipedia documents, creates a knowledge graph from the collected documents, and displays the answer to the query found in the relevant documents. The system is not limited by synonym words as it utilizes wordnet for similar word meanings. An example:

For the query "How did Micheal Jackson die?", the relevant graph edges would be:

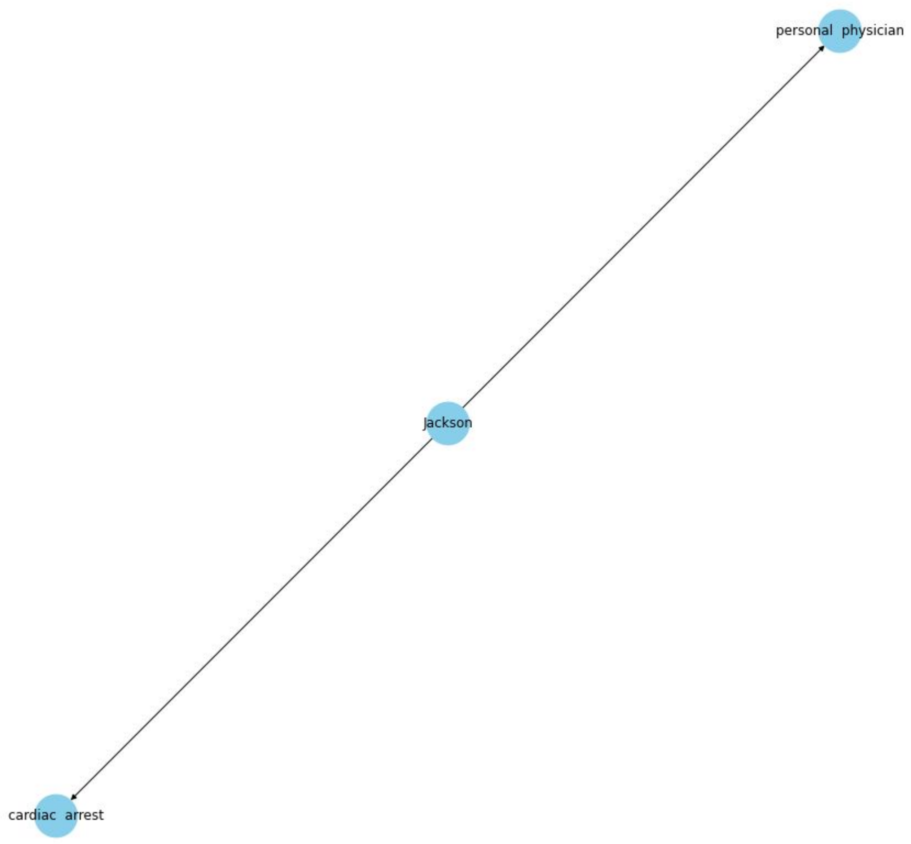


Figure 4.4 Graph edges for query "How did Micheal Jackson die?"

And this corresponds to the knowledge that Michael Jackson died of a cardiac arrest, and that his personal physician had a hand in the death. Another interesting insight is for the query "performed at" and its synonyms.

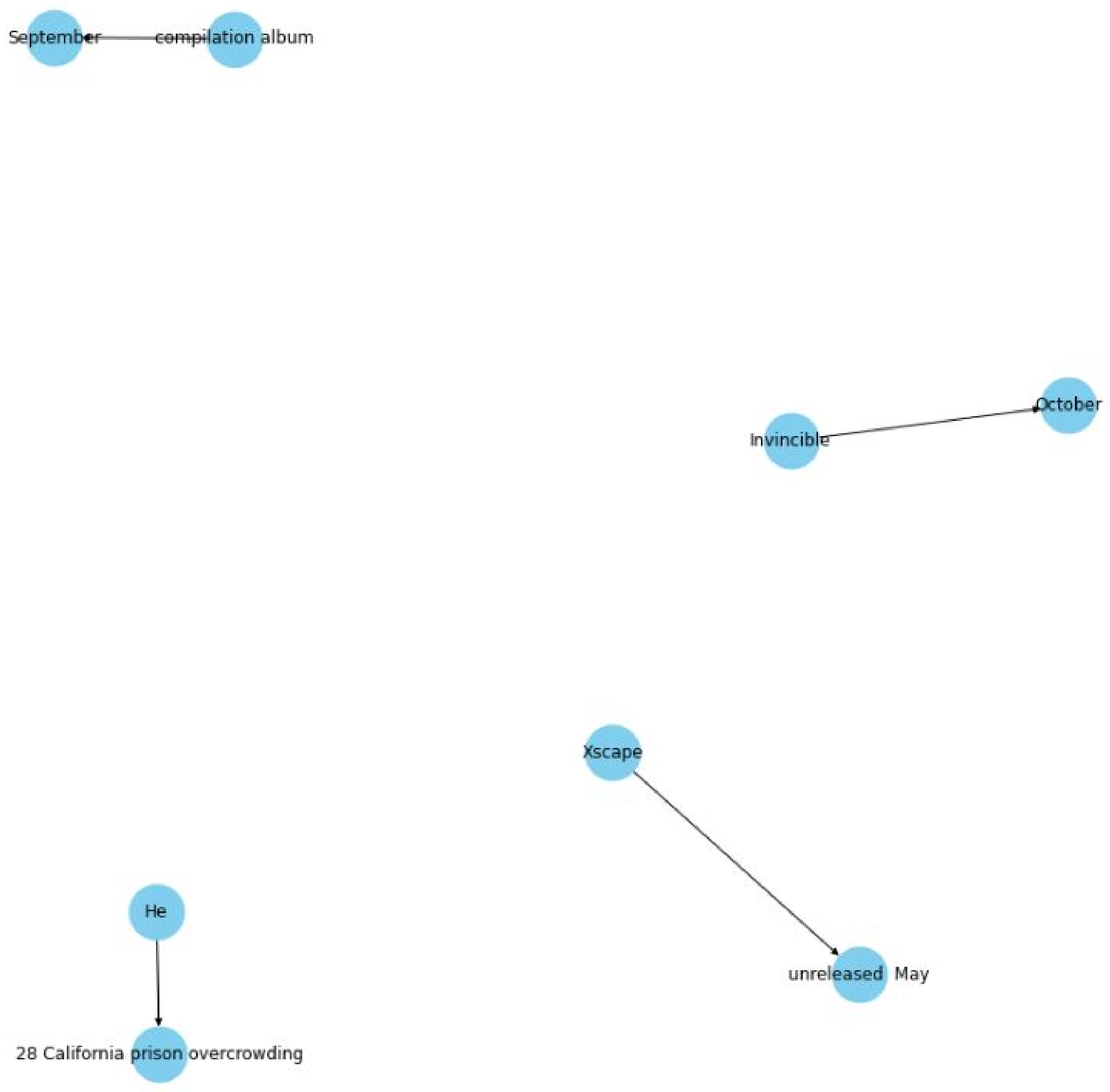


Figure 4.5 Graph edges for relation "perform"

Chapter 5

Conclusion

5.1 CONCLUSION

Knowledge graphs are a good way to parse and simplify text based documents into a network form that reduces storage space required and also makes searching for queries simple. They are especially useful for Question Answering systems as queries can also be reduced to entities that can be searched for in the graph as well as relations that form the edges of the graph. A dynamic approach to the formation of knowledge graphs based on web documents is optimal for real time applications.

5.2 FUTURE WORK

- Refining the NLP processes used in finding entities and relations to give wider results.
- Expand the graph from wikipedia to other infotainment webpages like: ask.com, and other article sites such as TOI, Buzzfeed etc.
- Increasing the scope into other languages such as regional Hindi, Tamil, Telugu etc. and international Spanish, French, German etc.
- A cache for the most recent/ most frequently searched page knowledge graphs can be created to reduce processing time.

References

[1] Diefenbach, D., Lopez, V., Singh, K. et al. Core techniques of question answering systems over knowledge bases: a survey. Knowl Inf Syst 55, 529–569 (2018).

[2] Bao, Junwei, et al. "Constraint-based question answering with knowledge graph." Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. 2016.

[3] Diefenbach, Dennis, et al. "Question answering benchmarks for wikidata." 2017.

[4] Zesch, Torsten, Christof Müller, and Iryna Gurevych. "Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary." LREC. Vol. 8. No. 2008. 2008.

[5] Buscaldi, Davide, and Paolo Rosso. "Mining knowledge from wikipedia for the question answering task." Proceedings of the international conference on language resources and evaluation. 2006.

[6] Chen, Danqi, et al. "Reading wikipedia to answer open-domain questions." arXiv preprint arXiv:1704.00051 (2017).

[7] Hakimov, Sherzod, et al. "Semantic question answering system over linked data using relational patterns." *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. 2013.

[8] Huang, Xiao, et al. "Knowledge graph embedding based question answering." Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. 2019.