NATURAL LANGUAGE PROCESSING

FALL SEMESTER 2020-21

TOPIC - Question Answering System

Team Members

18BCE2030	**Ramesh Sachan**
18BCE2037	**Aryaman Jain**
18BCE2026	**Ankit Yadav**
18BCE0693	**Vaani Tripathi**

Acknowledgement

I would like to thank our Natural Language Processing professor, Dr. Sharmila Banu K for her constant guidance and support without which this project would not have been possible. I would also like to thank our friends and family for their support which helped us finalize this project within the limited time frame.

Problem Description

Today the world is full of articles on a large variety of topics. We aimed to build a question-answering product that can understand the information in these articles and answer some simple questions related to those articles.

Proposed Solution

We plan to use Natural Language Processing techniques to extract the semantic & syntactic information from these articles and use them to find the closest answer to the user's question.

We'll extract NLP features like POS tags, lemmas, synonyms, hypernyms, meronyms, etc. for every sentence, and use the Apache Solr server to store & index all this information. We'll extract the same features from the question and form a Solr search query. This query will fetch the answer from the indexed Solr objects.

The main reason for using Solr is that Solr supports large-scale, distributed indexing, search, and aggregation/statistics operations, enabling it to handle applications large and small. Last but not the least out of the box ability to handle the synonyms or some other kind of simpler similarity of that kind out of the box.

Solr also supports real-time updates and can handle millions of writes per second. For instance, at Lucene/Solr Revolution, Salesforce shared that they have over 500 billion complex — not just logs — documents in Solr and are doing 7 billion updates per day with a sub 100-millisecond query latency. Based on a number of other talks at Revolution (Bloomberg, Microsoft, Wal-Mart, et. al.) as well as knowledge of my company's clientele, Salesforce isn't alone in those numbers.

Implementation Details

Minimum Hardware Requirements:

● Processor	:	Intel i5 7th Gen (to run Solr Server and Index the data onto it)
● RAM	:	8 GB
● Hard Disk Space	:	15 GB

Programming tools:

- Python (*version*: 3.8.6) - terminal
- Apache Solr (*version*: 8.6.3)
- NLTK library (*version*: 3.5)
- Spacy library (*version*: 2.3.2)
- en_core_web_sm & json & glob
- pysolr (3.9.0) (It is a lightweight Python client for Apache Solr. It provides an interface that queries the server and returns results based on the query.)

Architecture:

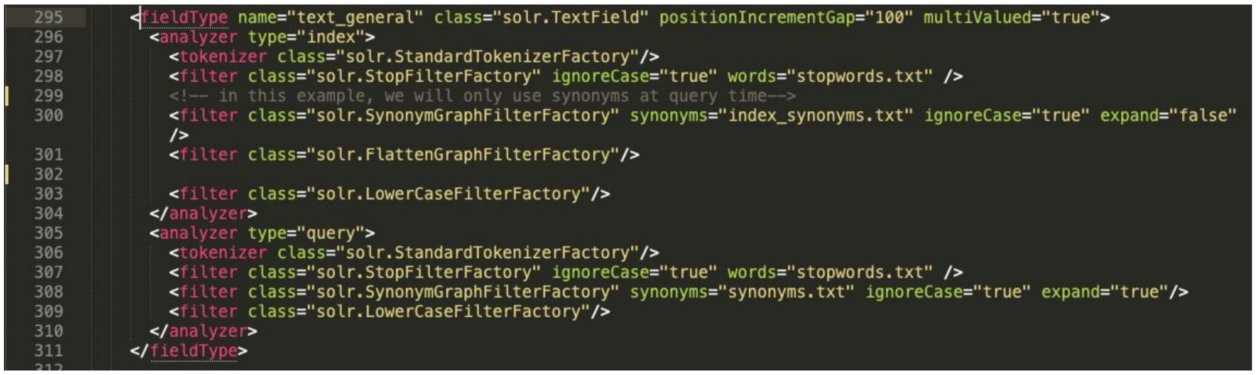
We divided the project into following 7 steps:

Step 1:

- Read all the sentences from the given corpus.
- Extract the following NLP features from each sentence – word tokens, lemmas, stems, synonyms, hypernyms, hyponyms, holonyms, meronyms, named entities, dependency parsed tree.
- We used Spacey library to create a dependency parsed tree of the sentence & stored it in a list.

Step 2:

- Send the sentence & it's extracted features to Apache Solr for indexing.
- Each indexed object is a list of key value pairs where each key is an NLP feature (ex. Synonyms, hypernyms, etc.) & its value is stored in csv format.
- Solr has an internal synonyms.txt file which accepts csv values of words that are not common & specific to our domain.
- Solr considers these values as synonyms when indexing and querying. Ex.: UTD, The University of Texas at Dallas, UT Dallas.
- This can be achieved by making a configuration change in the managed-schema file in Solr's directory as shown in figure 1.
- At the end of this step, the entire corpus would be indexed and stored in Solr, ready to handle the queries given to it in a proper format and then answering them.



```
<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100" multiValued="true">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
    <!-- in this example, we will only use synonyms at query time-->
    <filter class="solr.SynonymGraphFilterFactory" synonyms="index_synonyms.txt" ignoreCase="true" expand="false"
    />
    <filter class="solr.FlattenGraphFilterFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
    <filter class="solr.SynonymGraphFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
```

Figure 1: managed-schema.xml

Step 3:

- The program requires questions to be saved in a .txt file & it's path should be passed as a parameter while running the program.
- The questions can be of 3 types: **Who**, **When** and **Where**
- The type of question is used to determine the named-entity type of answer required NER (Named Entity Recognition).

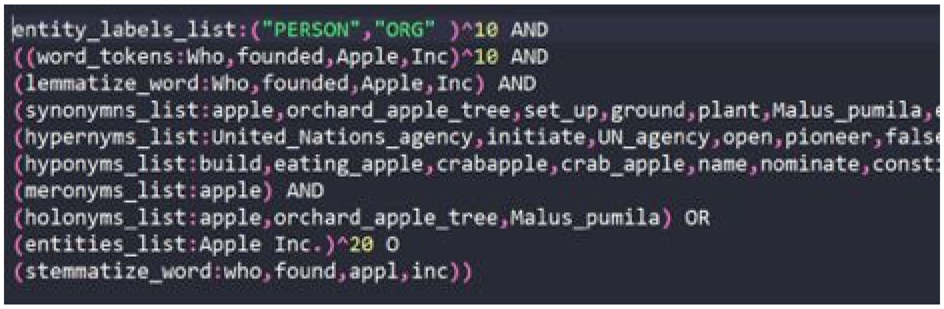
Question Type	Required NER type of Answer
Who	PERSON / ORG
When	TIME / DATE
Where	LOC / GPE

- NLP features are extracted for each question like in step 1.

Step 4:

- Solr accepts query in key-value format and it also supports logical operators like AND, OR.
- We create a concatenated query of the extracted NLP features from the question.
- The motive behind this step is simple - to create a query which will have a greater match score with the required sentence in Solr Index.
- By using NLP features we increase the chances of matching in cases where the exact word in the question doesn't occur in the sentence stored in Solr.
  Ex: If the question has a token: 'founded' but it's answer sentence has a token: 'established', the query would still be able to match them as they would be present in the synonyms list.
- Some features are more probable to give better matches and they are given preference over others by adding boosting weights to them.

- A sample query is shown in figure 2:



```
entity_labels_list:("PERSON","ORG" )^10 AND
((word_tokens:Who,founded,Apple,Inc)^10 AND
(lemmatize_word:Who,founded,Apple,Inc) AND
(synonymns_list:apple,orchard_apple_tree,set_up,ground,plant,Malus_pumila,
(hypernyms_list:United_Nations_agency,initiate,UN_agency,open,pioneer,fals
(hyponyms_list:build,eating_apple,crabapple,crab_apple,name,nominate,const
(meronyms_list:apple) AND
(holonyms_list:apple,orchard_apple_tree,Malus_pumila) OR
(entities_list:Apple Inc.)^20 O
(stemmatize_word:who,found,appl,inc))
```

Figure 2: Sample Query

Step 5:

- A connection is opened to Solr and the query is parsed which returns a list of Solr objects.
- These Solr objects contain the best possible matches that Solr found for the given query.
- They are arranged in the descending order of the match score which Solr handles internally.
- Every object contains the same features that were indexed in Step 2. This helps us to extract any information about these sentences without processing them further, thus saving computational time and resources.

Step 6:

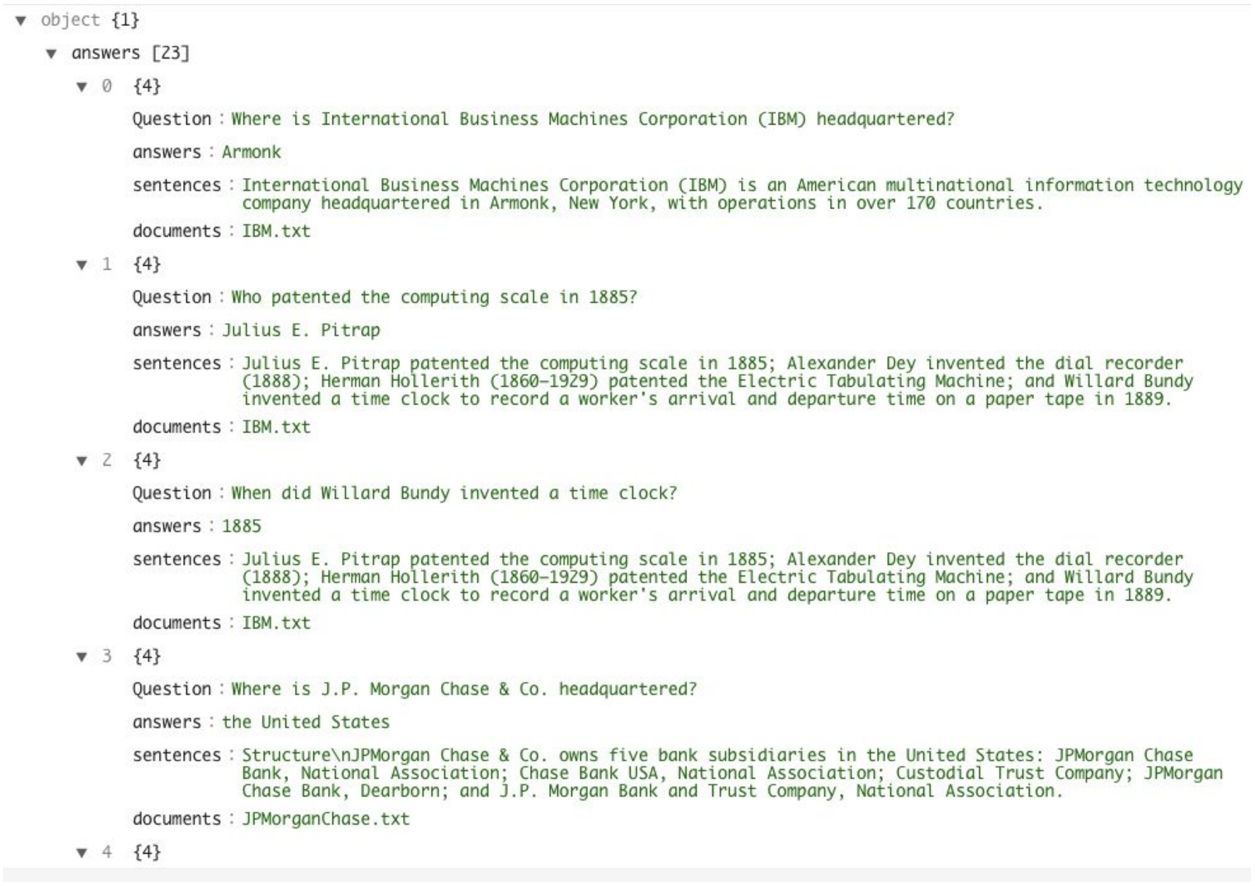
- Top 5 results for every search query are taken to extract the answer.
- Best possible sentence is selected from them using the Dependency Parsed tree present in the Solr object of the sentence.
- The required answer is extracted from the sentence using the dependency parsed tree tags & NERs. Ex: for WHEN questions, tokens with DATE or TIME tags are chosen.

Step7:

- The extracted results from Step 6 are stored in a JSON format as follows:

```
{
        "Question": "question string",
        "answers”:{
        //answers to question here
        },
        "sentences”:{
        //supporting sentences containing answers to question here
        },
        "documents”:{
        //supporting Wikipedia documents containing answers to question here
        }
}
```

- One JSON object is created for every question. They are saved in a JSON array & dumped into ‘answers.json’ file.
- Figure 3 shows a screenshot of answers.json

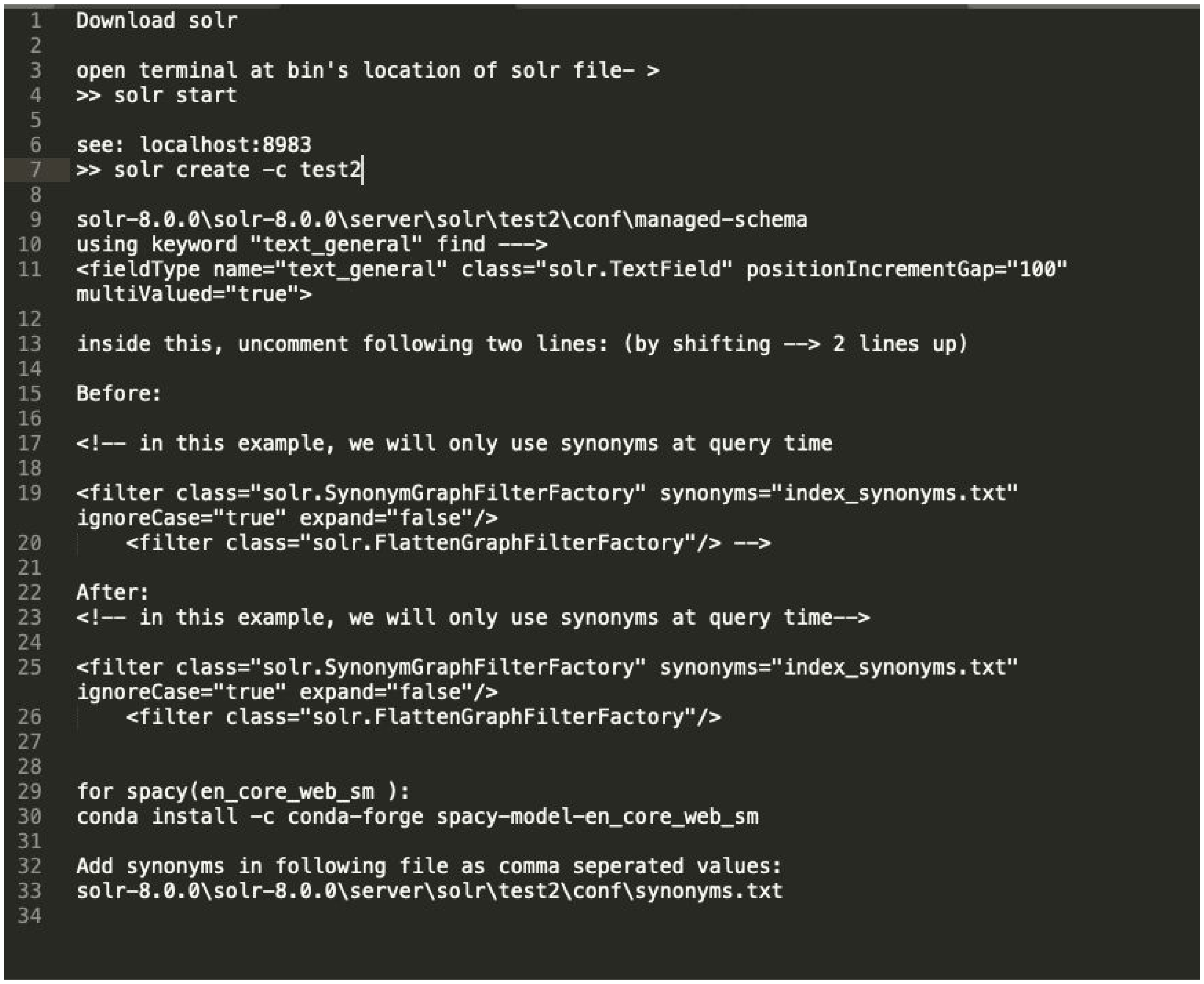


```
▼ object {1}
  ▼ answers [23]
    ▼ 0  {4}
         Question : Where is International Business Machines Corporation (IBM) headquartered?
         answers : Armonk
         sentences : International Business Machines Corporation (IBM) is an American multinational information technology
                     company headquartered in Armonk, New York, with operations in over 170 countries.
         documents : IBM.txt
    ▼ 1  {4}
         Question : Who patented the computing scale in 1885?
         answers : Julius E. Pitrap
         sentences : Julius E. Pitrap patented the computing scale in 1885; Alexander Dey invented the dial recorder
                     (1888); Herman Hollerith (1860-1929) patented the Electric Tabulating Machine; and Willard Bundy
                     invented a time clock to record a worker's arrival and departure time on a paper tape in 1889.
         documents : IBM.txt
    ▼ 2  {4}
         Question : When did Willard Bundy invented a time clock?
         answers : 1885
         sentences : Julius E. Pitrap patented the computing scale in 1885; Alexander Dey invented the dial recorder
                     (1888); Herman Hollerith (1860-1929) patented the Electric Tabulating Machine; and Willard Bundy
                     invented a time clock to record a worker's arrival and departure time on a paper tape in 1889.
         documents : IBM.txt
    ▼ 3  {4}
         Question : Where is J.P. Morgan Chase & Co. headquartered?
         answers : the United States
         sentences : Structure\nJPMorgan Chase & Co. owns five bank subsidiaries in the United States: JPMorgan Chase
                     Bank, National Association; Chase Bank USA, National Association; Custodial Trust Company; JPMorgan
                     Chase Bank, Dearborn; and J.P. Morgan Bank and Trust Company, National Association.
         documents : JPMorganChase.txt
    ▼ 4  {4}
```

Figure 3: Sample results in answer.json

Problems encountered:

- The main problem that I encountered while doing this project was setting up the solr server properly. For that I made a step by step guide to set up the server properly for future reference.



```
Download solr

open terminal at bin's location of solr file- >
>> solr start

see: localhost:8983
>> solr create -c test2

solr-8.0.0\solr-8.0.0\server\solr\test2\conf\managed-schema
using keyword "text_general" find --->
<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100"
multiValued="true">

inside this, uncomment following two lines: (by shifting --> 2 lines up)

Before:

<!-- in this example, we will only use synonyms at query time

<filter class="solr.SynonymGraphFilterFactory" synonyms="index_synonyms.txt"
ignoreCase="true" expand="false"/>
    <filter class="solr.FlattenGraphFilterFactory"/> -->

After:
<!-- in this example, we will only use synonyms at query time-->

<filter class="solr.SynonymGraphFilterFactory" synonyms="index_synonyms.txt"
ignoreCase="true" expand="false"/>
    <filter class="solr.FlattenGraphFilterFactory"/>


for spacy(en_core_web_sm ):
conda install -c conda-forge spacy-model-en_core_web_sm

Add synonyms in following file as comma seperated values:
solr-8.0.0\solr-8.0.0\server\solr\test2\conf\synonyms.txt
```

- Indexing each and every sentence took a long time when I tried to hit it to the Solr. I resolved it by indexing the entire document (list of sentences) at once.
- The words utd, The Univ of Texas at Dallas, The UT Dallas are used interchangeably in the questions or corpus. I resolved it by making these words synonyms in the synonyms.txt file. Also some of the additional synonyms that I could find that would be useful.
- Getting the required sentences in top 5 results in Solr was challenging. I resolved it by using the boosted weights for some of the few features.
- Refining the Solr query such that it selects the correct sentence from the list of Solr results could be the room for the improvement in this project.

Assignment Links

NLP Hands-on:

https://github.com/holps-7/NLP-weekly

Nat Geo Group Activity:

https://github.com/holps-7/CSE4022-Group5-Activity