International Journal of Research in Engineering, Science and Management
Volume 4, Issue 12, December 2021
https://www.ijresm.com | ISSN (Online): 2581-5792

Question Answering System Using Natural Language Processing

Atharva Mangeshkumar Agrawal[1*], Akhil Atri[2], Ayanesh Chowdhury[3], Rajeev Koneru[4], Kedareswara Abhinav Batchu[5], Sai Charan Reddy Mallavaram[6]

***Abstract*: Question Answering (QA) system in facts retrieval is a venture of mechanically answering an accurate answer to the questions requested by way of human in natural language the use of either a pre-structured database or a collection of natural language documents. It gives simplest the asked statistics as opposed to looking complete files like seek engine. As facts in everyday lifestyles is growing, to be able to retrieve the precise fragment of data even for a simple query calls for massive and high-priced assets. This is the paper which describes the distinctive technique and implementation information of question answering machine for popular language and additionally proposes the closed domain QA System for dealing with documents related to education acts sections to retrieve more specific answers using NLP strategies.**

***Keywords*: NLP, QA system.**

1. Introduction

Question answering is an essential NLP hassle and a long-status synthetic intelligence milestone. QA structures permit a person to specific a question in natural language and get a direct and brief reaction. QA systems are now determined in search engines like google and phone conversational interfaces, and that they're pretty top at answering easy snippets of statistics. On extra difficult questions, but, these commonly handiest cross as a long way as returning a list of snippets that we, the customers, need to then browse via to locate the answer to our query.

Reading comprehension is the capability to study a piece of textual content and then solution questions about it. Reading comprehension is tough for machines because it requires both herbal language information and knowledge of the world.

A. 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.

B. 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 primary purpose for the use of Solr is that Solr helps large-scale, disbursed indexing, seek, and aggregation/statistics operations, allowing it to deal with programs large and small. Last but no longer the least out of the container capacity to deal with the synonyms or a few different kind of easier similarity of that type out of the container.

Solr also helps actual-time updates and might manage millions of writes in line with second. For instance, at Lucene/Solr Revolution, Salesforce shared that they've over 500 billion complicated — now not just logs — documents in Solr and are doing 7 billion updates consistent with day with a sub 100-millisecond question latency. Based on some of other talks at Revolution (Bloomberg, Microsoft,

Wal-Mart, et. Al.) in addition to knowledge of my organization's clients, Salesforce isn't on my own in the ones numbers.

Fig. 1. managed-schema.xml

2. Implementation Details

A. Minimum Hardware Requirements

- Processor: Intel i5 7th Gen
- RAM: 8 GB
- Hard Disk Space:15 GB

B. 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)

*Corresponding author: agrawalatharva0777@gmail.com

- en_core_web_sm & json &glob
- pysolr (3.9.0) (It is a lightweight Python client for Apache Solr.

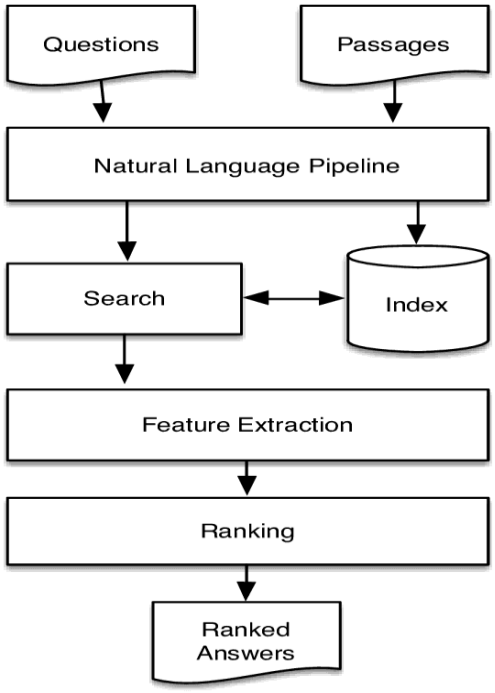

Fig. 2. Architecture

C. *Architecture*

We divided the venture into following 7 steps:

Step 1:

- Read all of the sentences from the given corpus.
- Extract the following NLP functions from every 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 & saved it in a list.

Step 2:

- Send the sentence & it's extracted capabilities to Apache Solr for indexing.
- Each listed object is a listing of key price pairs wherein every secret is an NLP characteristic (ex. Synonyms, hypernyms, and so on.) & its fee is stored in csv layout.
- Solr has an inner synonyms.Txt report which accepts csv values of words that aren't commonplace & precise to our area.
- Solr considers those values as synonyms when indexing and querying. Ex.: UTD, The University of Texas at Dallas, UT Dallas.
- This may be done via making a configuration alternate inside the managed-schema document in Solr's listing as proven in determine 1.
- At the stop of this step, the entire corpus would be indexed and saved in Solr, geared up to handle the queries given to it in a proper format after which answering them.

Step 3:

- The software calls for questions to be saved in a .Txt file & it's route should be handed as a parameter at the same time as running the program.
- The questions can be of 3 types: Who, When and Where.
- The form of query is used to determine the named-entity sort of solution required NER (Named Entity Recognition).

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

Step 4:

- Solr accepts question in key-value layout and it additionally helps logical operators like AND, OR.
- We create a concatenated question of the extracted NLP functions from the question.
- The cause behind this step is simple - to create a question that allows you to have a more match score with the specified sentence in Solr Index.
- By using NLP features we growth the possibilities of matching in instances wherein the precise phrase within the question doesn't arise within the sentence stored in Solr.
  Ex: If the query has a token: 'founded' but it's answer sentence has a token: 'installed', the query would nevertheless be capable of suit them as they might be gift in the synonyms listing.
- Some capabilities are extra in all likelihood to give better matches and they may be given desire over others by means of adding boosting weights to them.
- A sample question is proven in parent 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))
```

Fig. 3. Sample query

Step 5:

- A connection is opened to Solr and the query is parsed which returns a list of Solr items.
- These Solr gadgets include the pleasant possible matches that Solr located for the given question.
- They are organized in the descending order of the suit score which Solr handles internally.
- Every object contains the identical capabilities that have been indexed in Step 2. This allows us to extract any records approximately those sentences without processing them similarly, as a consequence saving computational time and

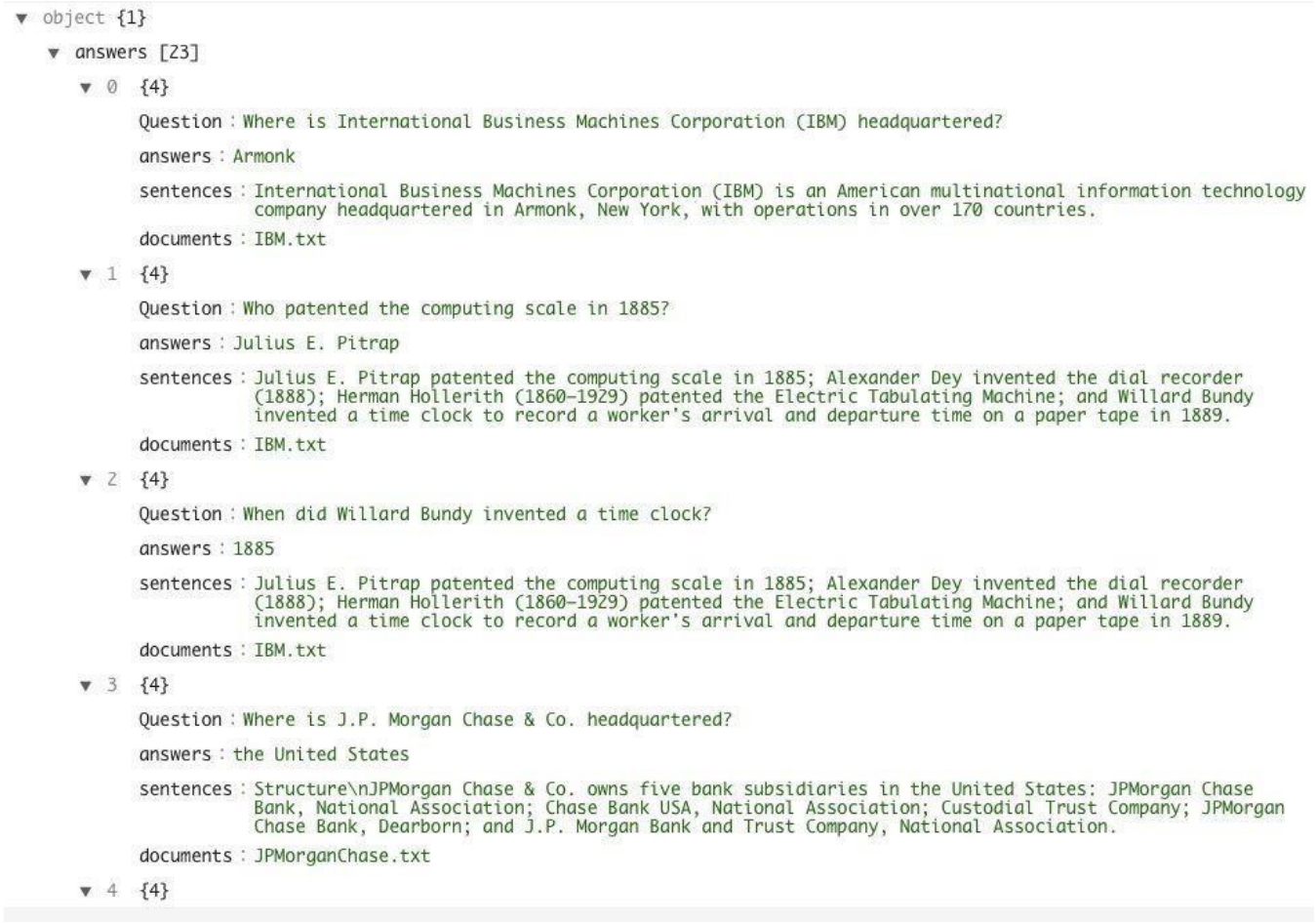

Fig. 4. answers.json

resources.

Step 6:

- Top five consequences for every seek question are taken to extract the answer.
- Best viable sentence is chosen from them using the Dependency Parsed tree gift in the Solr item of the sentence.
- The required answer is extracted from the sentence the usage of the dependency parsed tree tags & NERs. Ex: for WHEN questions, tokens with DATE or TIME tags are selected

Step 7:

- 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 4 shows a screenshot of answers.json

In the making of the NLP Features.py module.which is the step 2 and the step 3

- Tokenization of words
- POS (part of speech) tagging
- Extracting Synonyms, Hypernyms, Hyponyms, Holonyms and Meronyms as a list for further indexing.
- Returning NER (named entity recognition) to the main.py module Also contributed in making a function to write /append data to the JSON file.
- Entities that are in the answer.json file are the question , its answer , the sentence from which the answer is extracted and the document it refered to extract the answer

Problems encountered(standard):

- The phrases utd, The Univ of Texas at Dallas, The UT Dallas are used interchangeably within the questions or corpus. We resolved it by means of making those phrases synonyms inside the synonyms.Txt file. Also some of the extra synonyms that we may want to discover that would be useful.
- Getting the required sentences in pinnacle 5

consequences in Solr turned into challenging. We resolved it by the use of the boosted weights for few features (entities, word_tokens and required_entities)

3. Conclusion

How these troubles had been resolved:

- First we tried to research all the simple concepts and them implemented it.
- The documentation of NLTK turned into of lots assist.
- Geeks for geeks
- mistakes related to code have been resolved via mutual discussions, stackover waft, github, youtube.

References

[1] https://en.wikipedia.org/wiki/Question_answering#:~:text=Question%20answering%20(QA)%20is%20a,humans%20in%20a%20natural%20language.