Question Answering System

Project Report

Team Name: Entropy

Team Members:

Shanjida Khatun (sxk200130@utdallas.edu)
Allen Wang (axw161630@utdallas.edu)

CS6320.501: Natural Language Processing
Professor: Dr. Mithun Balakrishna

Table of Contents

Contents **Page**

Problem Description

Question Answering (QA) is a critical NLP problem and a long-standing artificial intelligence milestone. QA systems allow a user to express a question in natural language and get an immediate and brief response. QA systems are now found in search engines and phone conversational interfaces, and they are good at answering simple snippets of information. On more hard questions, however, these normally only go as far as returning a list of snippets that we must then browse through to find the answer to our question.

Reading comprehension is the ability to read a piece of text and then answer questions about it. Reading comprehension is difficult for machines because it requires both natural language understanding and knowledge of the world.

The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset made up of questions posed by crowd workers on a collection of Wikipedia articles, with the response to each question being a text segment, or span, from the relevant reading passage, or the question being unanswerable.

The reading sections in SQuAD are taken from high-quality Wikipedia pages, and they cover a wide range of topics from music celebrities to abstract notions. A paragraph from an article is called a passage, and it can be any length. Reading comprehension questions are included with each passage in SQuAD. These questions are based on the passage's content and can be answered by reading it again. Finally, we have one or more answers to each question.

One of SQuAD's distinguishing features is that the answers to all the questions are text portions, or spans, in the chapter. These can be a single word or a group of words, and they are not limited to entities–any range is fair game.

Our goal is to build a question-answering product that can understand the information in these articles and answer some simple questions related to those articles. Our purpose is to locate the text for any new question that has been addressed, as well as the context. This is a closed dataset, so the answer to a query is always a part of the context, and that the context spans a continuous span.

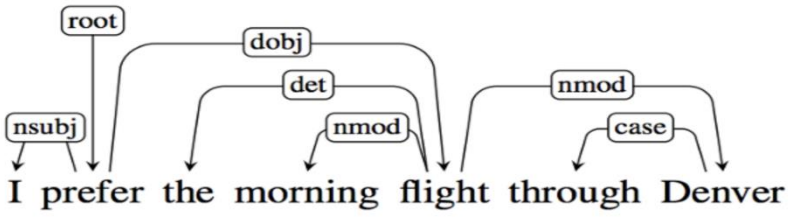
Proposed Solution

We plan to use Natural Language Processing techniques to extract the semantic and syntactic information from these articles and use them to find the closest answer to the user's question. We will extract NLP features like tokenize text, POS tags, lemmas, dependency parsing, hypernyms, hyponyms, holonyms, meronyms, etc. for every sentence, and use an Apache Solr tool to store and index all this information. We will extract the same features from the question and form an Apache Solr search query. This query will fetch the answer from the indexed Solr objects.

Sentence Segmentation: We have used Doc.has_annotation with the attribute name "SENT_START" to see if a Doc has sentence boundaries. Here the paragraph is broken into a meaningful sentence. Spacy can produce random sentences based on the period, it executes intelligent splitting. A screenshot of using sentence segmentation in our code is shown below:

```
#Finding Named Entity
entities = []
entity_labels = []
nlp = en_core_web_sm.load()
doc = nlp(sentence)
for name in doc.ents:
    entities.append(name.text)
    entity_labels.append(name.label_)
```

Parsing Dependencies: The "Dependency Parse Tree" is another feature we have used to solve some problems. The model's accuracy will improve because of this. Spacy tree parsing was used since it has a robust API for traversing through the tree. Below the text, directed, named arcs from heads to dependents show the relationships between the words. Because we generate the labels from a predefined inventory of grammatical relations, we call this a Typed Dependency structure. It also comprises a root node, which denotes the tree's root, as well as the entire structure's head. An example of a dependency parse tree is shown below:



Full Implementation Details

Programming Tools

- Python: *version*: 3.8, Spyder
- Apache Solr: *version*: 8.0
- NLTK library: *version*: 3.2.5
- Spacy library: *version*: 2.0.13

Architectural Diagram

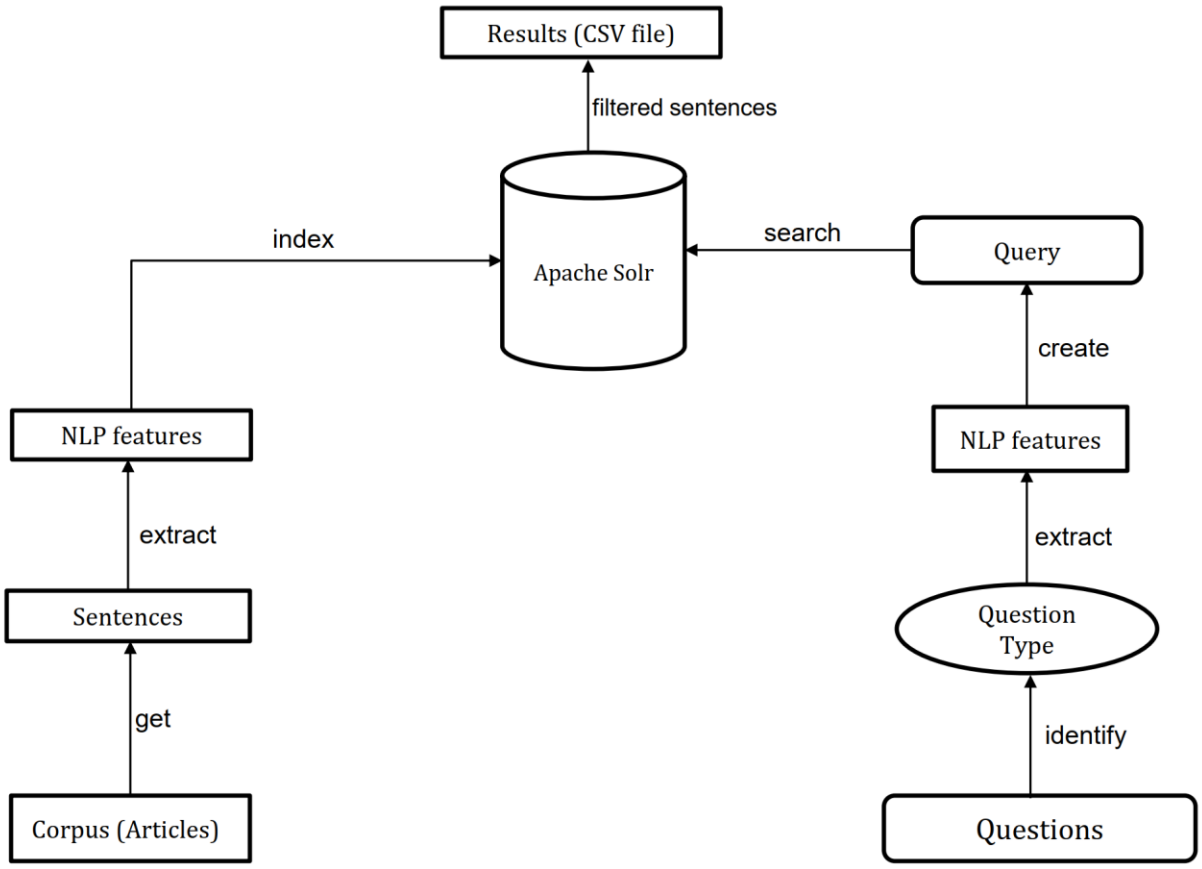


Figure 1: *A full Architectural Diagram of the QA System.*

Implementation Details

Feature Extraction:

Given articles, read all the sentences. NLP features extracted from the sentences. Features included word tokens, lemmas, synonyms, hypernyms, hyponyms, holonyms, meronyms, stems, POS tags, dependency parse tree, named entities, the root of a sentence. Spacey library allowed us to create a dependency parse tree of the sentence.

A screenshot of NLP features extractions:


```
{'id': '109_3',
'words': ['It', 'occurs', 'mainly', 'northern', 'hemisphere', 'birds', 'funnelled', 'specific', 'routes', 'natural', 'barriers', 'Mediterranean', 'Sea', 'Caribbean', 'Sea'],
'sentence': 'It occurs mainly in the northern hemisphere, where birds are funnelled on to specific routes by natural barriers such as the Mediterranean Sea or the Caribbean Sea.',
'lemmas': ['I', 't', 'o', 'c', 'c', 'u', 'r', 's', 'm', 'a', 'i', 'n', 'l', 'y', 'n', 'o', 'r', 't', 'h', 'e', 'r', 'n', 'h', 'e', 'm', 'i', 's', 'p', 'h', 'e', 'r', 'e', 'b', 'i',
'stems': ['I', 't', 'o', 'c', 'c', 'u', 'r', 'm', 'a', 'i', 'n', 'l', 'i', 'n', 'o', 'r', 't', 'h', 'e', 'r', 'n', 'h', 'e', 'm', 'i', 's', 'p', 'h', 'e', 'r', 'b', 'i', 'r', 'd',
'tags': [('It', 'PRP'), ('occurs', 'VBZ'), ('mainly', 'RB'), ('in', 'IN'), ('the', 'DT'), ('northern', 'JJ'), ('hemisphere,', 'NN'), ('where', 'WRB'), ('birds', 'NNS'), ('are', 'VBP
'parse_tree': [['nsubj', 'occurs', 'It'], ['ROOT', 'occurs', 'occurs'], ['advmod', 'occurs', 'mainly'], ['prep', 'occurs', 'in'], ['det', 'hemisphere', 'the'], ['amod', 'hemisphere'
'hypernyms': ['engineering', 'engineering_science', 'applied_science', 'technology', 'become', 'appear', 'come_along', 'Middle_English', 'geographical_area', 'geographic_area', 'geo
            te_quantity', 'large_indefinite_amount', 'turbulent_flow', 'body_of_water', 'water', 'large_indefinite_quantity', 'large_indefinite_amount', 'turbulent_flow'],
'hyponyms': ['anticipate', 'arise', 'come_up', 'backfire', 'backlash', 'recoil', 'befall', 'bechance', 'betide', 'break', 'break', 'recrudesce', 'develop', 'chance', 'come', 'come_a
            ', 'cock', 'coraciiform_bird', 'cuculiform_bird', 'dickeybird', 'dickey-bird', 'dickybird', 'dicky-bird', 'gallinaceous_bird', 'gallinacean', 'hen', 'Ibero-mesornis', 'r
            ive', 'private_road', 'highway', 'main_road', 'line', 'railway_line', 'rail_line', 'post_road', 'roadway', 'shortcut', 'cutoff', 'crosscut', 'side_road', 'skid_road',
'meronyms': ['air_sac', 'beak', 'bill', 'neb', 'nib', 'pecker', "bird's_foot", 'bird', 'fowl', 'feather', 'plume', 'plumage', 'furcula', 'hindquarters', 'croup', 'croupe', 'rump',
           ', 'recess', 'Caribbean_Island', 'Antigua_and_Barbuda', 'Bahamas', 'Commonwealth_of_the_Bahamas', 'Bahama_Islands', 'Barbados', 'Cuba', 'Republic_of_Cuba', 'Dominica',
'holonyms': ['Earth', 'earth', 'world', 'globe', 'cerebrum', 'bird', 'hydrosphere', 'hydrosphere'],
'rootOfSentence': 'occurs',
'synonymns': ['information_technology', 'IT', 'happen', 'hap', 'go_on', 'pass_off', 'occur', 'pass', 'fall_out', 'come_about', 'take_place', 'occur', 'come', 'occur', 'chiefly', 'pr
             'raw', 'rude', 'natural', 'natural', 'born', 'innate', 'lifelike', 'natural', 'barrier', 'barrier', 'roadblock', 'barrier', 'Mediterranean', 'Mediterranean_Sea', 'Medit
'entities': ['the Mediterranean Sea', 'the Caribbean Sea'],
'entity_labels': ['LOC', 'LOC']}
```


Figure 2: *NLP features extractions.*

Indexing in Apache Solr:

All sentences and their features are sent to Apache Solr for indexing. Indexed objects are added as a list of key-value pairs of all the extracted NLP features. Solr has an internal synonyms.txt file that accepts CSV values of words that are not common and specific to our domain.

A screenshot of the synonyms.txt file:

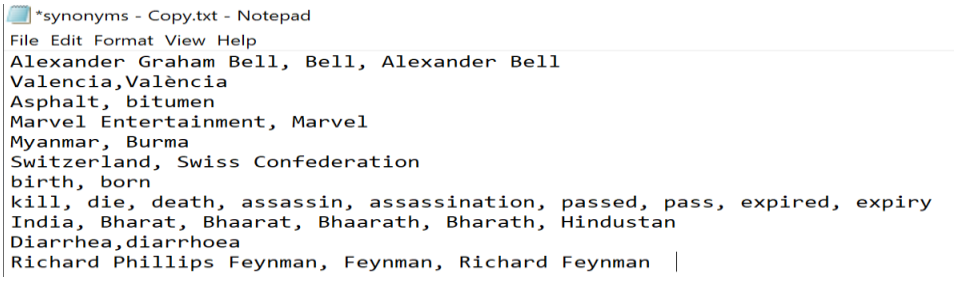


Figure 3: *The synonym.txt file.*

Solr considers these values as synonyms when indexing and querying. Example: Bell, Alexander Graham Bell, Alexander Bell. This can be achieved by making a configuration change in the managed-schema file in Solr's directory as shown in figure 4.

```
<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100" multiValued="true">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.SynonymGraphFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms.txt"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
    <filter class="solr.SynonymGraphFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms.txt"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
```

Figure 4: *The managed-schema.xml.*

The entire corpus will be indexed and stored in Solr ready to be queried for answers. Since there are lots of documents and lots of features for each, it takes time for indexing all the articles. Figure 5 shows a screenshot of some features on an indexed sentence in Apache Solr:


```
"id":"109_0",
"words":["Bird",
  "migration",
  "regular",
  "seasonal",
  "movement",
  "often",
  "north",
  "south",
  "along",
  "flyway",
  "breeding",
  "wintering",
  "grounds"],
"sentence":["Bird migration is the regular seasonal movement, often north and south along a flyway, between breeding and wintering grounds."],
"lemmas":["B",
  "i",
  "r",
  "d",
  "m",
  "i",
  "g",
  "r",
  "a",
  "t",
  "i",
  "o",
  "n",
  "r",
  "e",
  "g",
  "u",
  "l",
  "a",
  "r",
```


Figure 5: *Solr Object.*

Questions Format:

The program requires questions to be in a .txt file format and its path should be passed as a parameter while running the program. The questions are of 3 types: Who, When, and What. For answering the 'Who' and the 'When' type of questions, we have used the Named Entity Recognition technique. NLP features are extracted for each question as we did in the initial step.

Question type	Named Entity Recognition (NER) Type
Who	PERSON, ORG
When	TIME, DATE

Querying:

Solr accepts queries 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 motivation is to create a query that will have a greater match score with the required sentence in Solr. 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.

Example: If the question has token: 'founded' but its answer sentence has 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. We tried various combinations of boosting and found required NERs, word tokens, and NERs gave the best results when boosted.

A sample query is shown in figure 6:

```
query = "entity_labels:("+",".join(entity_type)+" )^20 AND "((words:"+words+")^20 OR \
        (synonymns:"+synonymns+")^10 OR (hypernyms:"+hypernyms+") OR (hyponyms:"+hyponyms+") OR \
        (meronyms:"+meronyms+") OR (holonyms:"+holonyms+") OR (rootOfSentence:"+rootOfSentence+") OR \
        (stems:"+stems+")^10 AND (entities:"+entities+")^20 OR (lemmas:"+ lemmas+")^10)"
```

***Figure 6: A** sample query in Solr.*

Result Extraction:

The extracted answers from the querying stage are stored in a CSV format as shown below:

Question_string_1, article_id_1, answer_sentence_1
Question_string_2, article_id_2, answer_sentence_2
……
Question_string_N, article_id_N, answer_sentence_N

The results are stored in a CSV file named "result.csv" on a per-row basis with each element in its own column. If the file already exists, then our program will keep what is already in the CSV file and simply append the new results.

Results and Error Analysis

Results:

Figure 7 shows a screenshot of our results:

	A	B	C
1	Question	Article_	Answer
2	Who mediated the truce with Khomeini?	400	Subsequently, Khomeini accepted a truce mediated by the UN.
3	When did an empire collapse after Alexander's c	56	At the age of 19, he wrote a report on his work and sent it to philologist Alexander Ellis, a colleague of his father (who would late
4	What is the Leader of the Revolution also knowr	NA	Answer not found
5	What is the nickname for Tucson?	390	Roughly 150 Tucson companies are involved in the design and manufacture of optics and optoelectronics systems, earning Tucsc
6	Who sold Arizona?	390	The state manages all water in Arizona through its Arizona Department of Water Resources (ADWR).
7	When was Arizona purchased from Mexico?	390	Arizona, south of the Gila River was legally bought from Mexico in the Gadsden Purchase on June 8, 1854.
8	What type of fuel is used by Fajr-3 missile?	400	The Fajr-3 (MIRV) is currently Iran's most advanced ballistic missile, it is a liquid fuel missile with an undisclosed range which was
9	Who succeeded Reza Shah?	400	In 1941, Reza Shah was forced to abdicate in favor of his son, Mohammad Reza Pahlavi, and established the Persian Corridor, a r
10	What led to students capturing the US embassy?	400	Sanctions have led to a steep fall in the value of the rial, and as of April 2013 one US dollar is worth 36,000 rial, compared with 1
11	Who is the Supreme Leader?	400	The Assembly elects the Supreme Leader and has the constitutional authority to remove the Supreme Leader from power at any
12	What distance can the Fajr-3 missile travel?	400	The Fajr-3 (MIRV) is currently Iran's most advanced ballistic missile, it is a liquid fuel missile with an undisclosed range which was

Figure 7: *A sample results in result.csv.*

Analysis: In the above document, there are 11 questions. According to the complex level (1, 2, 3), there are 4 questions (2-5) that are in complex level-1 (CL1), 5 questions (3-9) that are in complex level-2 (CL2), and 2 questions (11-12) that are in complex level-3 (CL3). We are getting 6 (2 CL1, 3 CL2, 1 CL3) correct results or sentences out of 11 correct sentences. We are getting 9 (2 CL1, 5 CL2, 2 CL3) correct document-id out of 11 document-id.

For one question, our question answering system didn't find the correct answer which returned "Answer not found". For four questions, it was not able to get the required sentences from the top 5 results in Solr.

We have found three correct answers that are in complex level 2 and one correct answer that is in complex level 3. We made it possible by adding synonyms and Named Entities in the query.

A Summary of the Problems Encountered and how these issues were solved

Indexing Time: Indexing took a long time when we tried to hit Solr for each, and every sentence. We resolved it by indexing the entire document (list of sentences) at once. Another approach to resolve this issue is if we set a variable named commit (parameter) as false (value), it took less time for indexing the features.

Synonyms: The words Bell, Alexander Graham Bell, Alexander Bell are used interchangeably in the questions or corpus. We resolved it by making these words synonyms in the synonyms.txt file.

Getting Correct Sentence: Getting the required sentences in the top 5 results in Solr was challenging. We resolved it by using the Named Entity Recognition technique for answering the 'who' and 'when' type questions.

We also resolve it by using the boosted weights for a few features such as words, lemmas, stems, synonyms, entities, and entity_type.

We have also extracted a root of a sentence from the dataset and question for improving the accuracy. By adding root in a query, sometimes it gives us correct answers for some typical questions.

Pending Issues

- We can get the required sentence as the first result of the Solr search query for any question.
- Accuracy can be improved in selecting the correct sentence from the list of Solr results.
- We can extract the correct answer from the result sentence if it has more than one NER.

Potential Improvements

- Pronoun resolution could be a potential improvement to extract an answer from the sentences which have pronouns in the places of subject or object.
- Refining the Solr query such that it selects the correct sentence from the list of Solr results.
- If it is possible to extract semantic relation during features extraction for both dataset and questions, then it will be a huge improvement for the system.
- If we set a threshold during document selection to determine how many documents represented the best compromise, then we can evaluate documents by comparing their identifier with those contained in the list.