Implementation of Question Answering System

NLP Project

Priyank Shah (pks170030)
Meetika Sharma (mxs173530)

Project Description:

The goal of this project is to implement a Question Answering system on factoid questions. Dataset contains Wikipedia articles to search for the answer of given question.

Proposed Solution

Our approach for Question Answering system is information retrieval based. We are indexing the Wikipedia articles by tokenizing into sentences and adding information helpful in efficiently and accurately retrieving such as dependency parse, POS tag, synonyms etc. Then, we are creating a search query based on the question to search in the indexed data. To create a search query from the question, we add features such as dependency parse, synonyms, stem, lemma etc.

Implementation Details:

- ### Feature Extraction:

We need to extract all the features from the provided Wikipedia articles. The system performs the following steps to extract the features:

- It tokenizes the articles into sentences and then into words.
- It extracts the lemma for each word. Lemma is the base form or dictionary form of the word.
- It also extracts the POS tag feature, which determines the part of speech for every word.
- For each sentence, the system also does dependency parsing, which helps in knowing the relation between words which occurs in the same sentence.
- WordNet provides with a list of synonyms, hypernyms, hyponyms. The system extracts these types of features for each word and store it in a dictionary.

Extract the features for all the sentences in all the documents. Index each word and above listed features as separate search fields in SOLR.

- ### Question processing:

The first task is to get the type of the question. There are 3 main types of the question for this project.

1) WHO
2) WHERE
3) WHEN

The system searches in the question if these ‘wh’ words are present and assigns the question type to the question accordingly. We also understand that WHO type of question will have

answer tagged by named entity recognition as PERSON or ORGANIZATION and WHEN type of question will have DATE and TIME and WHERE type of question will have STATE_OR_PROVINCE, CITY, COUNTRY, LOCATION.

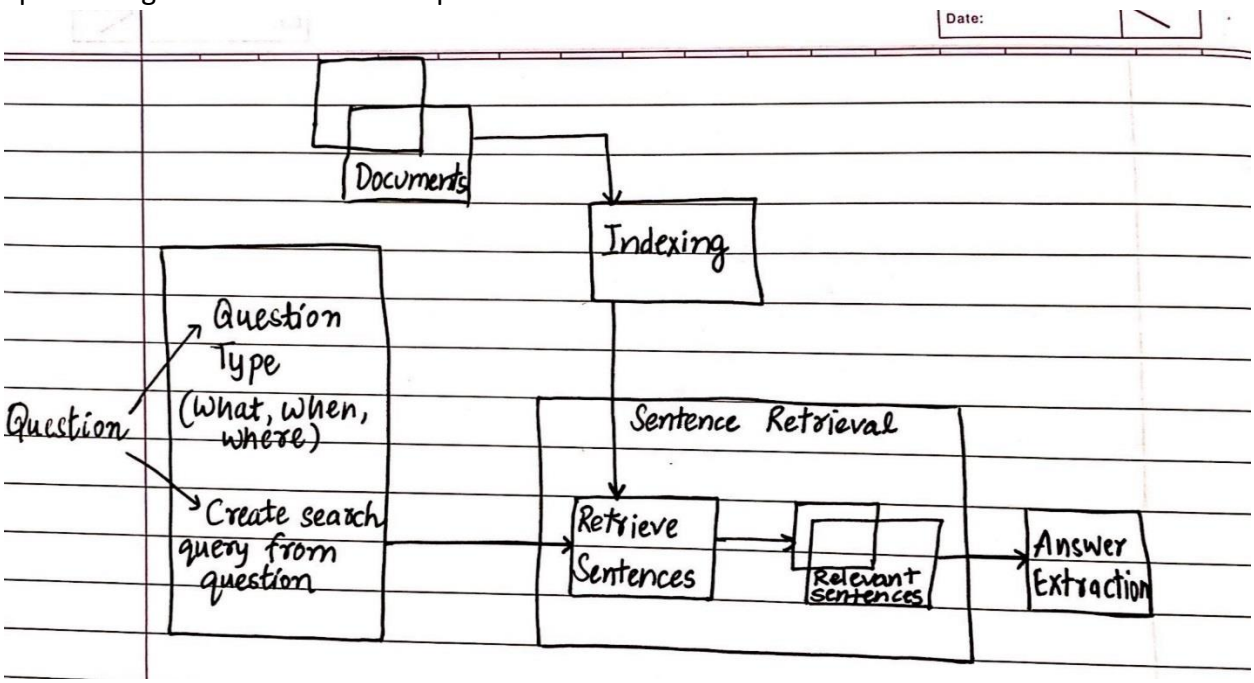
Task 1:

- Implementation of a deeper pipeline which extracts many features of the sentence.
- We are given a list of Wikipedia articles which consists of multiple sentences.
- We are also given a questions file which consists of multiple questions.
- The task here is to extract the features from each sentence and each question.
- For every article, we are first sentence tokenizing it into multiple sentences.
- On each sentence we perform word tokenize to get the list of words in the corresponding sentence.
- We find the lemma of each word. E.g. Lemma of studies in study
- We extract the part of speech tag for each word. E.g. POS Tag for shot is ('shot', 'VBD')
- We perform the dependency parsing for each sentence. The dependency parse tree helps us in knowing the important relations between words.
- This can be very useful to find the subject, verb, object of the sentence.
- We also extracted other features like hypernyms, hyponyms, meronyms, holonyms and synonyms of each word. As it is possible that question and answer does not contain the exact same word.
- These features will help us in extracting the top sentences which can be the answer for the given question.

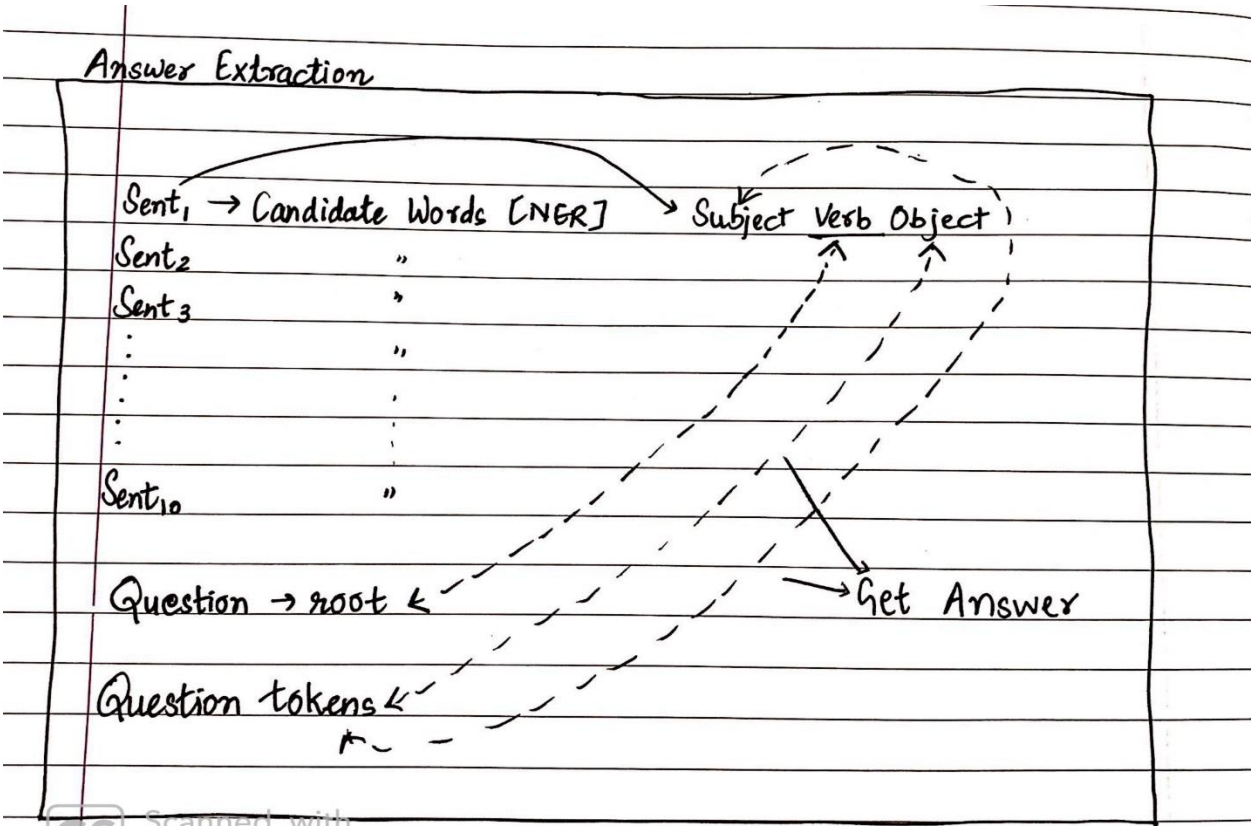
Result of task 1 for different text files has been stored in separate files in json format. Output is in the form of list of dictionaries. A sentence has key as token, lemma, POS Tag, dependency parse, synonyms, hypernymns, hyponyms, meronyms, holonyms and synonyms and value as their values.

Architectural Diagram:

1. Pipeline to get the answer of the question



2. Answer Extraction:



Task 2:

1. Solr has been used in this project for indexing the sentences. First, solr server is created using "createSolrSchema" function. In solr, we have "core" which stores our indexed data. We delete any earlier indexed data before creating a new index.
2. Then, we iterate over all the input data (Wikipedia articles) and sentence tokenize the file. Then, we iterate over the sentences and add them to solr.
3. To add a sentence to the solr core, we extract useful information from the sentence and add them to the solr core as well for indexing. We first word tokenize the sentence and extract its Pos Tag, lemma, Dependency parse, synonyms, hypernyms, hyponyms, meronyms, holonyms. We put all this information in a dictionary to be added into the solr core for indexing.
4. Next, we are reading all the questions from the question input file one by one and transforming a question in the searchQuery that we will use for searching in solr index.
5. To create the searchQuery, we perform similar steps as we did for the input sentences. We identify its type i.e. whether it is a who or when or where question. We are also extracting its tokens, Pos tag, lemma, dependency parse, synonyms, hypernyms, hyponyms, meronyms, holonyms and then we add all these values to the search query as a key-value pair.

Extracting answer from the list of sentences returned by solr:

- We have got a list of sentences after performing task 2. These are our candidate sentences for the answer. Now we need to find a sentence that will correctly answer out question.
- First, we are listing all possible answers from each candidate sentence based on Named Entity Recognition i.e. we are extracting candidate words which are: any word tagged as LOCATION for where type question or tagged as HUMAN or ORGANISATION for who type question.
- Then, we are extracting the root of the question using spacy.
- Next, we check if the root of the question and verb of the sentence are same or are in each other's synonyms. If they are, then we match their subject and object and check if the subject or object is in candidate words of that sentence. Then we return if it matches.

Problems encountered during the project:

1. We tried using whoosh for indexing, but it was not time efficient.
2. We were adding synonyms etc. in the text only and then index the modified data however it was not giving good results. Then, we added synonyms etc. in the schema of solr instead of adding that in the text and we got better results.
3. There are text files having ANSI encoding that we had to convert to UTF-8 encoding.
4. WordnetLemmatizer does not give correct results. Then we provided Pos tag as well to the lemmatizer and got better results.
5. Lesk also gives wrong "best sense" despite giving the context.
6. There are a lot of synonyms for every word and search results were not good after we added synonyms as a parameter in the search query.
7. There are abbreviations used in questions that are not defined in wordnet and are not listed as synonyms. Therefore, indexing does not give correct results for such questions E.g. Question is “Who founded UTD”. However, the answer contains UT Dallas or University of Texas at Dallas. Hence, we do not get correct results.
8. We are using Stanford core NLP’s Openie to extract subject, verb and object of the sentence but it was unable to find subject, verb and object of the question. It gives None as subject, verb and object of the question. To overcome this problem, we are finding the root of the question using spacy.
9. There are questions we do not have exact words as the answer sentence. Hence, we are searching in the synonyms of those words, but synonyms list is very long and gives random words as the synonym. Therefore, boosting the impact of synonym in search does not give correct results.
10. We faced difficulty in finding the query string for best search results. We tried different combinations and some combination work for some questions but not others. We have customized query string to get correct results for most questions.

Task 3:

Executable of python has been made using “pyinstaller”. All of the code has been put in a single python file for the conversion to executable.

Command to run pyinstaller:

pip install pyinstaller

pyinstaller --onefile NLProject.py

where NLProject.py is the name of the python program.

Requirements:

Language: Python

Python Libraries used:

1. pysolr - used to index data i.e wikipedia articles
2. nltk - used for feature extraction e.g. synonyms, tokens
3. json - to use json modules to load and dump json data into files
4. subprocess - to run command prompt commands from the python program
5. Wordnet - lemmatizer is used to extract lemmas.
6. Spacy - to extract the relation in a sentence
7. pyinstaller - to convert python program into exe

Servers used:

1. Stanford CoreNLP - port 9000
2. Stanford CoreNLPDependencyParse - port 9000
3. Solr Server - port 8983

References :

1. https://medium.com/dreamcatcher-its-blog/making-an-stand-alone-executable-from-a-python-script-using-pyinstaller-d1df9170e263
2. https://lucene.apache.org/solr/guide/
3. Speech and Language Processing - Third Edition, Daniel Jurafsky, James H. Martin