TEAM PROJECT

Done by: Kruthika Chakka (kxc200005) and Kiran Raj Devraj (kxr190038)

Problem description

Given a set of articles, use NLP concepts and algorithms to find answers to a set of given questions. For the project, we need to implement a Question Answering system using NLP features and techniques for the following what, who and when questions. These questions can be straight forward or convoluted based on the level of difficulty but, all answers will be found in the aforementioned articles. The data is extracted from Stanford Question Answering Dataset (SQuAD), a reading comprehension dataset, consisting of questions posed by crowd-workers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable.

The QA system is required to return the sentence containing the answer for the input question from the given 30 articles.

• Input: A file containing natural language questions one per line

• Output: supporting sentence which contains the answer for each question, and the article id which contains the supporting answer sentence

No Machine Learning (either traditional or Neural Network) techniques can be used for extracting the relevant answer sentence.

The entire project is divided into the following tasks:

Task 1: Implement a deeper NLP pipeline to extract at least the following N LP based features from the articles in the dataset and natural language questions:

- Tokenize text into sentences and words
- Lemmatize the words to extract lemmas as features
- Part-of-speech (POS) tag the words to extract POS tag features
- Perform dependency parsing or full syntactic parsing to parse tree based patterns as features
- Using WordNet, extract hypernymns, hyponyms, meronyms, AND holonyms as features

Task 2: Implement a QA system to extract relevant sentence(s) for a natural language question from the processed SQuAD dataset:

- Run the above described deeper NLP on the data set and extract NLP features
- Run the above described deeper NLP on the natural language question and extract NLP features
- Implement a NLP knowledge driven (i.e. template, statistical, heuristic/rule, or a combination) approach to extract the relevant answer sentence for a natural language question from the given 30 articles dataset.

Task 3: Provide an executable program that will accept input and produce output in the form of a CSV file

Proposed solution

Approach:

Perform task 1 separately form the Q&A system using the following steps:

- Read all the articles from “articles” folder and store the document text into memory.
- Using the nltk package do the following operations on the articles:
    - Split articles into individual sentences
    - POS tag each sentence using the nltk sentence tagger
    - Perform dependency tagging on the POS tagged sentences
    - Split sentences into individual words/ tokens
    - Find lemmas of each word
    - Find hypernymns, hyponyms, meronyms, and holonyms for each word using wordnet

Perform task 2 & task 3 by implementing the following steps:

- Read the questions from “questions.txt” file and find answers one at a time
- Pre-process the questions to generate multiple version of each question by doing the following:
    - Replace the verbs in the question with its synonyms to generate different versions of the question which have the same meaning and hence the same answer
- Compare the proper nouns and common nouns in the question with the proper nouns and common nouns in each sentence of the data corpus. If the sentence from corpus satisfies the following condition, filter it out and add it to possible answers list:
    - The proper nouns in the question and in the sentence from corpus should match
    - The count of common nouns in the question and the count of common nouns in the sentence from corpus should match
- Do this for each version of the question
- With the possible answers list, run a cosine similarity between each possible answer and the question. Do this for all versions of the question.
- The possible answer with the highest cosine similarity with any version of the question is final answer for that question

Repeat the above process for all questions

By performing the following steps on the data corpus, we were able to build the Q&A system described above:

- Read the questions from the text file “questions.txt”
- Read all the articles from the folder “articles”
- Split the articles into sentences using the nltk package
- POS tag all the words in each of the sentences
- Split each of the above sentences into words using the nltk package
- Do the past 3 steps for the questions as well
- Remove all the stop words and symbols from the sentences in the corpus and questions
- Find the proper nouns, common nouns and verbs in the questions
- For each question, do the following:
    - Find different versions of the question:
        - Find all the verbs in the question
        - Find all synonyms for each verb in the question

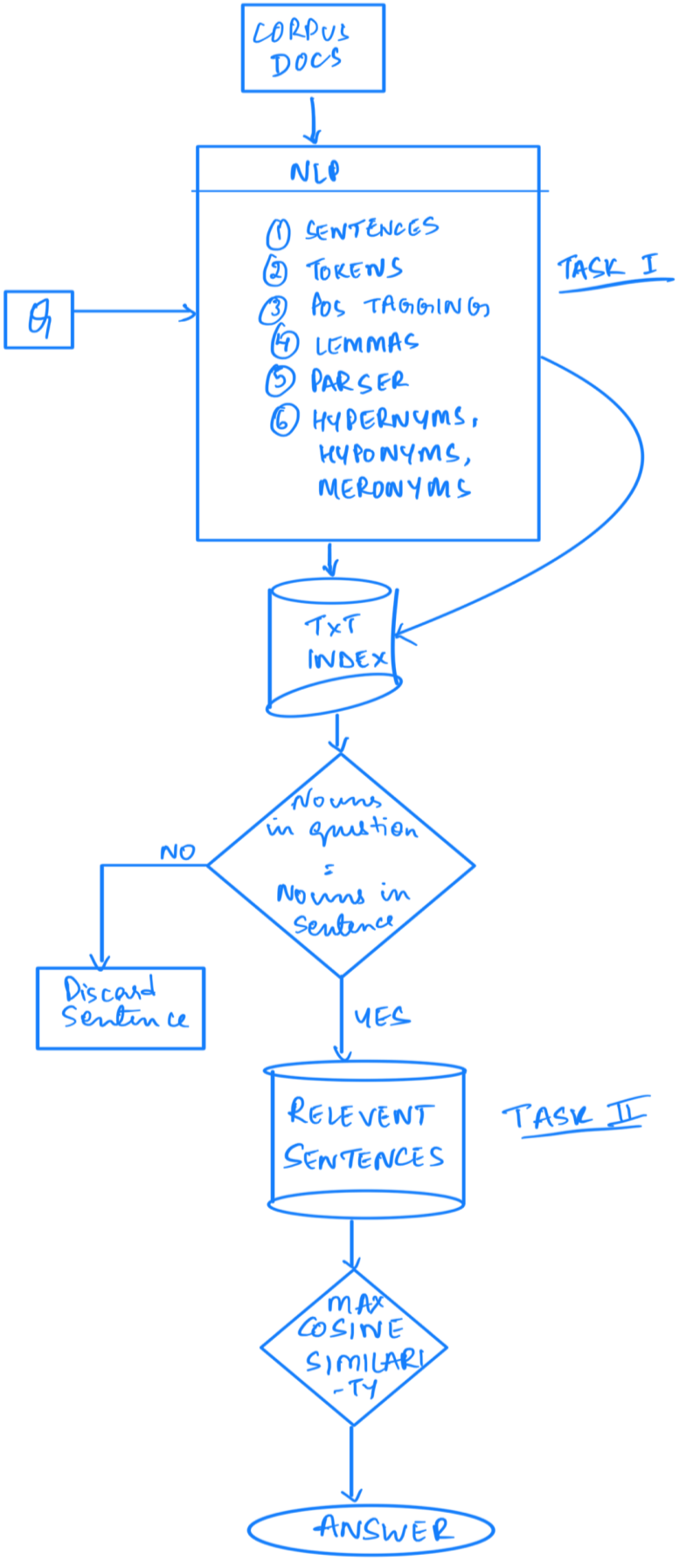
        - Create different versions of the question by replacing the verb with each of its synonym
    - Filter the sentences using the following conditions :
        - Proper nouns in the question should match the proper nouns in the sentence
        - Count of common nouns in the question should match the count of common nouns in sentence.
        - Do this for every version of the question
    - Using the filtered sentences and different versions of the question, find the cosine similarity between each version – sentence pair.
    - The filtered sentence with the highest cosine similarity is the answer.

Full implementation details

Programming tools (including third party software tools used)

- Since we are using the "ntlk" package in our project, the following commands should be run before running our projects to download the appropriate packages:
    - `import nltk`
    - `nltk.download('punkt')`
    - `nltk.download('averaged_perceptron_tagger')`
    - `nltk.download('wordnet')`
    - `nltk.download('stopwords')`
- Also, you need to download the Stanford parser package from the URL given below:
    - https://nlp.stanford.edu/software/lex-parser.shtml#Download
    - Click on "Download Stanford Parser version 4.2.0" button and save the zip file in the same location as the project.
    - Extract the contents of the zip file
- Download the core NLP package from:
    - https://stanfordnlp.github.io/CoreNLP/
    - Click on "download CoreNLP 4.3.2"
    - Save zip file in the same location as the project
    - Extract the contents of the zip file
- Download spacy using:
    - `Conda install -c conda-forge spacy` (in anaconda spyder environment)
- Download additional spacy packages as specified below:
    - Download a tar file from git-hub to avoid OS read error by doing:
      `pip install https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-2.3.1/en_core_web_sm-2.3.1.tar.gz`
    - `pip install –U pyresparser`
    - `pip install -U pip setuptools wheel`
    - `python -m spacy download en_core_web_sm`
    - Open the anaconda prompt as administrator and type the following command to doenload the English parsing package: `python -m spacy download en`

Architectural diagram



Results and error analysis (with appropriate examples)

Results: We are able to produce correct answers for 70% of the questions of a mixture difficulties (type 1, type 2 and type 3). This is done using just the proper nouns and common nouns in the question.

Examples:

- Question: “What is the nickname for Tucson?"

  Answer: "Roughly 150 Tucson companies are involved in the design and manufacture of optics and optoelectronics systems, earning Tucson the nickname Optics Valley."

  Model generated answer: “Tucson was awarded a gold rating for bicycle-friendliness by the League of American Bicyclists in 2006”

  Analysis: The key components in the question which can be used to find the answer are “Tuscon” and “nickname”. However, nickname is getting tagged as a “verb”. So, even though the answer sentence gets filtered as one of the possible sentences using the proper noun “Tuscon” that can be the answer, it loses out while finding the cosine similarity.

- Question: "Who sold Arizona?"

  Answer: "Arizona, south of the Gila River was legally bought from Mexico in the Gadsden Purchase on June 8, 1854."

  Model generated answer: “On Sentinel Peak (also known as "'A' Mountain"), just west of downtown, there is a giant "A" in honor of the University of Arizona.”

  Analysis: The key words that link the question to the answer is “sold” and “bought”. “Sold” is an antonym of “bought”. So, the answer uses the antonym to categorize the sentence as answer. But, we do not use antonyms in our analysis

A summary of the problems encountered during the project and how these issues were resolved

Problems:

- We were unable to use solr/ elastic search to index the documents.
  - Solution: we created list of lists which held the following data for each document
    - Text in document
    - Sentences in document
    - Word in document
    - POS tagged sentences
    - Lemmas of each word according to its POS tag
  - We used this list of lists to build functions and run the above described algorithm
- Using tf-idf calculation between each document and question, we were trying to rank the documents according to the probability that it might have the answer. But, due to prepositions and common nouns, a lot of unelated documents were ranked before the document which had the correct answer.
  - Solution: We filtered all the sentences in all documents using proper nouns and count of common nouns.

- Using the filtered sentences, we found the cosine similarity between then and the different versions of the question
- The filtered sentence with highest cosine similarity is the answer

Pending issues

- Use antonyms appropriately: As describes in the example above, the answer sometimes contains the antonym of the key word of the question which makes it the correct answer.
  Eg. Sold – bought
- Use verbs in the question efficiently: If the same verb repeats twice in the question, we find its synonyms and replace the verb with synonyms to form different versions of the question. So when we replace the verb with synonym – 1, we replace all occurrences of the verb. Instead of this, we can use different synonyms at different instances
- Use meonyms, hypernyms and hyponyms while forming different versions of the question

Potential improvements

- Change the code structure to use verbs, hypernyms, hyponyms and meronyms to filter potential answers for the questions.
  - Tweak the cosine similarity logic according to the feature used
- Use dependency parsing/ full parsing to find relations between words in the sentences and find similar relations in questions to match question – answer pair.