CS 6320: Natural Language Processing

The University of Texas, Dallas

-- Spring 2020 – Prof Dr. Mithun Balakrishna

Final Project - Information Extraction

Group Name: The Watchers

Members: 1) Anirudh Reddy Kaveti (axk190081)

2) Sai Krishna Prasad Kurapati (sxk190015)

Report Index

1) Problem Description:

The goal is to implement an Information Extraction Application using NLP features and techniques. Here the input is of different articles related to Organizations, Persons and Locations. From the given articles we have to generate the Information templates such as BUY, WORK and PART (Example: City, part of Country). The information may be in the form of different syntactic structures in the articles for which we have to traverse through each of the sentences by applying the semantic knowledge and various techniques to extract the templates. In addition to this task, NLP pipeline also needs to implemented in order to extract features for any given text document or sentence.

Template 1:

BUY (Buyer, Item, Price, Quantity, Source)

Template 2:

WORK (Person, Organization, Position, Location)

Template 3:

PART (Location, Location)

The output should be of JSON file which contains the format of sentences where the respective template has been found and its respective required entities and info.

Output Sample:

Document: *Amazon_com.txt*

Sentence(s)*: In 2017, Amazon acquired Whole Foods Market for US$13.4 billion, which vastly increased Amazon's presence as a brick-and-mortar retailer.*

Extracted Template*: BUY("Amazon", "Whole Foods Market", "US$13.7 billion", "", "")*

2) Solution Proposed:

Task 1:

For the first task to extract the features from the given text of document, our solution includes use of the available NLP libraries which were put forth in a good efficient way to generate the desired results. Here we have first tokenised the text document into sentences, later took each sentence and tokenised into words. Later we have used nltk library to do lemmatization, to generate pos tags, to generate features such as Hypernyms, Hyponyms, Meronyms and Holonyms. Using Spacy library, we have done dependency parsing.

Functions used:

I. Extracting Sentences – tokenizer.tokenize(content)
II. Word tokenization – nltk.word_tokenize(sentence)
III. Lemmatization – WordNetLemmatizer() → lemmatizer.lemmatize(word)
IV. Pos tagging – nltk.pos_tag(wordsList)
V. Synsets Extraction - wordnet.synsets(word)
VI. Hypernyms – extractedsynset.hypernyms() , Hyponyms - extractedsynset.hyponyms()
Meronyms – extrasynset.part_meronyms(), Hyponyms - extrasynset.part_meronyms()
VII. Dependency Parsing - nlp(sentence) → eachtok.text , eachtok.dep_ , eachtok.pos_

Task 2:

Template 1:

BUY (Buyer, Item, Price, Quantity, Source)

For this template we have collected all the possible synonyms for the buy word and have stored it as list of words. Initially after extracting the text data from the given document we have passed it through the pipeline which undergoes coreference resolution and the resolved text is generated.

The pipeline has been created in the following way - Initially used spacy library's neuralcoref to get the resolved text after coreference resolution, later extracted the sentences using nltk library with sentence tokenization function. Later traversed each sentence to undergo dependency parsing to generate the dependency and POS tags for the respective words which are key members in generating the template required.

Finally, we check the **dependency tag** of each token, whether it is root or verb followed by the check whether the word is present in the list of synonyms for 'Buy' which has been created initially. If it is present then the sentence will be sent to the template generation logic. And the final output is formatted as JSON file and will be displayed.

Template generation logic:

- Check the sentence is in active voice or passive voice. If the sentence has 'nsubjpass' token dependency path, then the sentence is in passive voice.
- If the sentence is in passive voice, the 'subject' of the sentence becomes the 'item' bought. The 'object' of the sentence is the 'buyer'.
- If the sentence is in the active voice, the 'subject' is the 'buyer' and the 'object' with the root of the sentence is the 'item'.
- To find the cost , we use named-entity recognition and find the entity of type 'MONEY'.
- To find the quantity, we use parts of speech tag and word with tag 'NUM' is choosen as quantity.

Template 2:

WORK (Person, Organization, Position, Location)

For this the text generation is similar to that of the template 1 till coreference resolution. But here we create a custom list of entities related to that of the work and we create a pipeline in such a way that we add this entity list to the NLP pipeline which helps us in generating the NER , POS tagging and dependency parsing more relevant to the template which we needed to generate. Later we tokenize the sentence using the spacy nlp library to generate entity labels and also respective dependency, pos tags.

Finally, we check whether any token contains the **entity** label as 'Job-Title' and also the sentence tokens contain another **entity** label as 'ORG', if both the tokens are present then the sentence will be sent to the template generation. And the final output is formatted as JSON file and will be displayed.

Template generation logic:

- First 'Job-Title' entity is generated from the training corpus and using entity recognition of Stanford core-NLP library.

- There are two patterns for this type of template, one is (Person, Job-title, Organisation) or (Job-title, Organisation, Person)
- If the sentence is in first pattern, we choose Person as the word which is 'subject' and also with entity type 'PERSON'. Job title and organisation are chosen using named entity recognition of space.
- For the second pattern, Job-title is the subject of the sentence and person is chosen based on the entity type using named entity recognition.
- Location of the organisation is chose if the entity type is 'GPE' and is related to organisation in the relation.
- If multiple persons or multiple job roles in different organisation are found, then those are stored as new relation.

Template 3:

PART (Location, Location)

Here also the text generation is similar to that of the template 1 till coreference resolution. Later using the spacy library we send this sentence through the nlp pipeline to generate entity labels as well as the respective dependency and pos tags.

Finally, we check whether any token contains the **entity** label as 'GPE' or ;LOC' and also the sentence tokens contain another **entity** label as 'LOC' or 'GPE', if the entity labels are greater than or equal to two then the sentence will be sent to the pattern generation logic which has been written. And the final output is formatted as JSON file and will be displayed

Template generation logic:

- For all the words in the list ['of', 'in', 'is'], all their children in the dependency parse tree are iterated and those with entity type of location are chosen.
- While selecting the children any compound places are concatenated to form the correct location.
- Now the final location list is iterated and relation is formed with the location in the left and location in the right in the original sentence.

3)Programming Tools:

Programming Language:

- Python 3.6

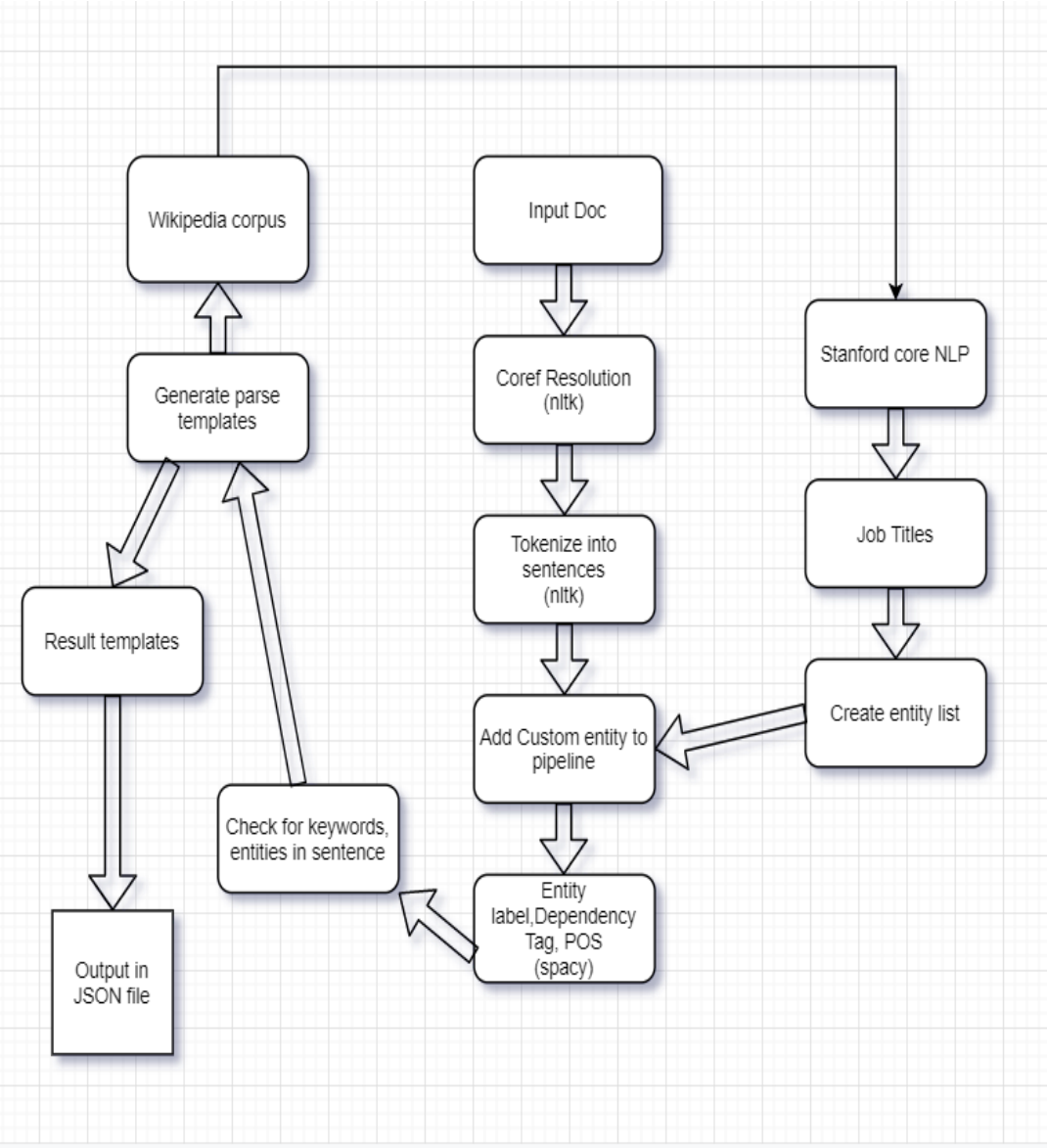
Tools used:

- Google Colab
- Visual Studio Code

Libraries used:

1) Nltk
    - Sentence_tokenize
    - Word_tokenize
    - Wordnet lemmatizer
    - Pos_tag
    - Wordnet Synset
        - Hypernyms and Hyponyms
        - Meronyms and Holonyms
2) Spacy
3) Spacy – LookUp
4) Python Collections
5) Pandas

4)Architectural Diagram:



5)Error Result Analysis and Resolution:

1) Job Title Extraction:

While performing the Work template extraction in the task 2. We have faced difficulty in finding the job titles from the sentence. Initially we have tried spacy ner to find the job title but it didn't recognize the Job title.

Resolution: We have used the Stanford core nlp to get the job titles and we have stored the results in the csv file which we have later used the file to recognize the job titles from text while generating the templates

2) Coreference Resolution:

We have encountered problem while finding the subjects in the templates where we got pronouns instead of proper nouns or nouns

Resolution: We have used neuralcoref library and extracted the resolved text, further used it as the main text to generate the templates.

6)Pending Issues:

1) We were not able to exactly recognize some of the states
2) We were not able to generate templates for the sentences which are bit complex. Template are majorly working for generic and most common sentences we encounter.

7)Potential Improvements:

1) More complex patterns recognition by implementation of regex
2) We could write custom ner for job titles instead of hard coding it.

8)References:

1) https://spacy.io/api/dependencyparser
2) https://www.nltk.org/py-modindex.html
3) https://stanfordnlp.github.io/CoreNLP/simple.html