Project Report

CS6320 - Natural Language Processing

Team InfoTractors

Spring 2021

Aakash Sunil (axs200020)

Kaushik Nadimpalli (kxn160430)

Problem Description

The purpose of this project is to implement a deep NLP Information Extraction application. Information Extraction (IE) is a very popular NLP problem in which we are trying to recognize entities and finding relationships between different text entities. Some popular applications of Information Extraction include Recommender Systems, Search Engines, Digital Libraries etc. Text can often be unstructured or structured. Extracting information from these types of text can often require the use of popular NLP algorithms, techniques, and libraries.

Our goal is to implement an end-to-end pipeline that analyzes features from a set of input articles, and then extracts specific types of templates from those articles in an expected json format. The templates we are concerned with for the scope of this project are:

Template 1: BORN(Person/Organization, Date, Location)

Template 2: ACQUIRE(Organization, Organization, Date)

Template 3: PART_OF(Organization, Organization)

PART_OF(Location, Location)

Problem Statement

“Given a set of input documents, our application will extract the required and additional features utilizing a deep NLP Pipeline, and then use those aforementioned features to extract the 3 templates in the desired output format.”

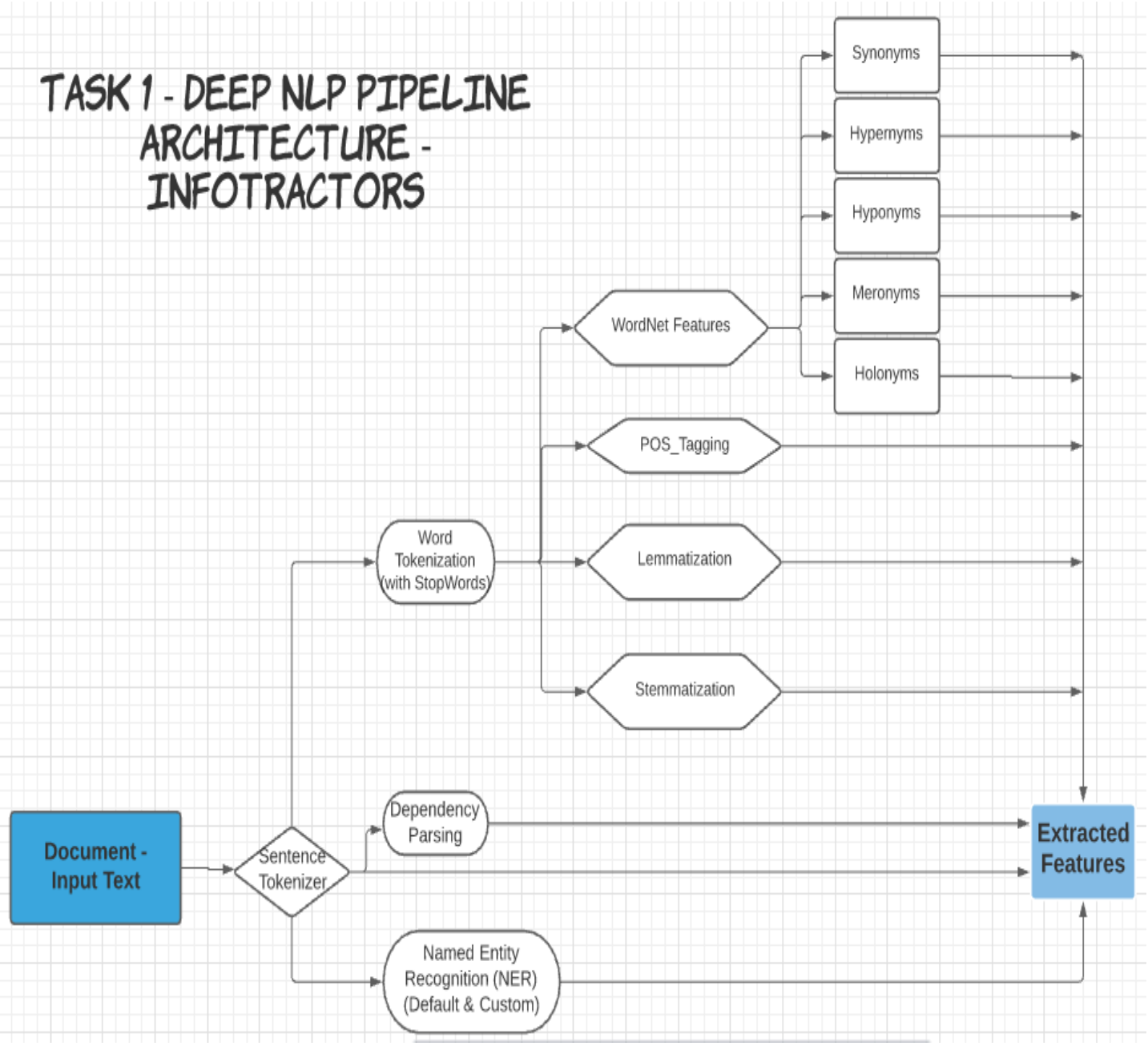
Proposed Solution

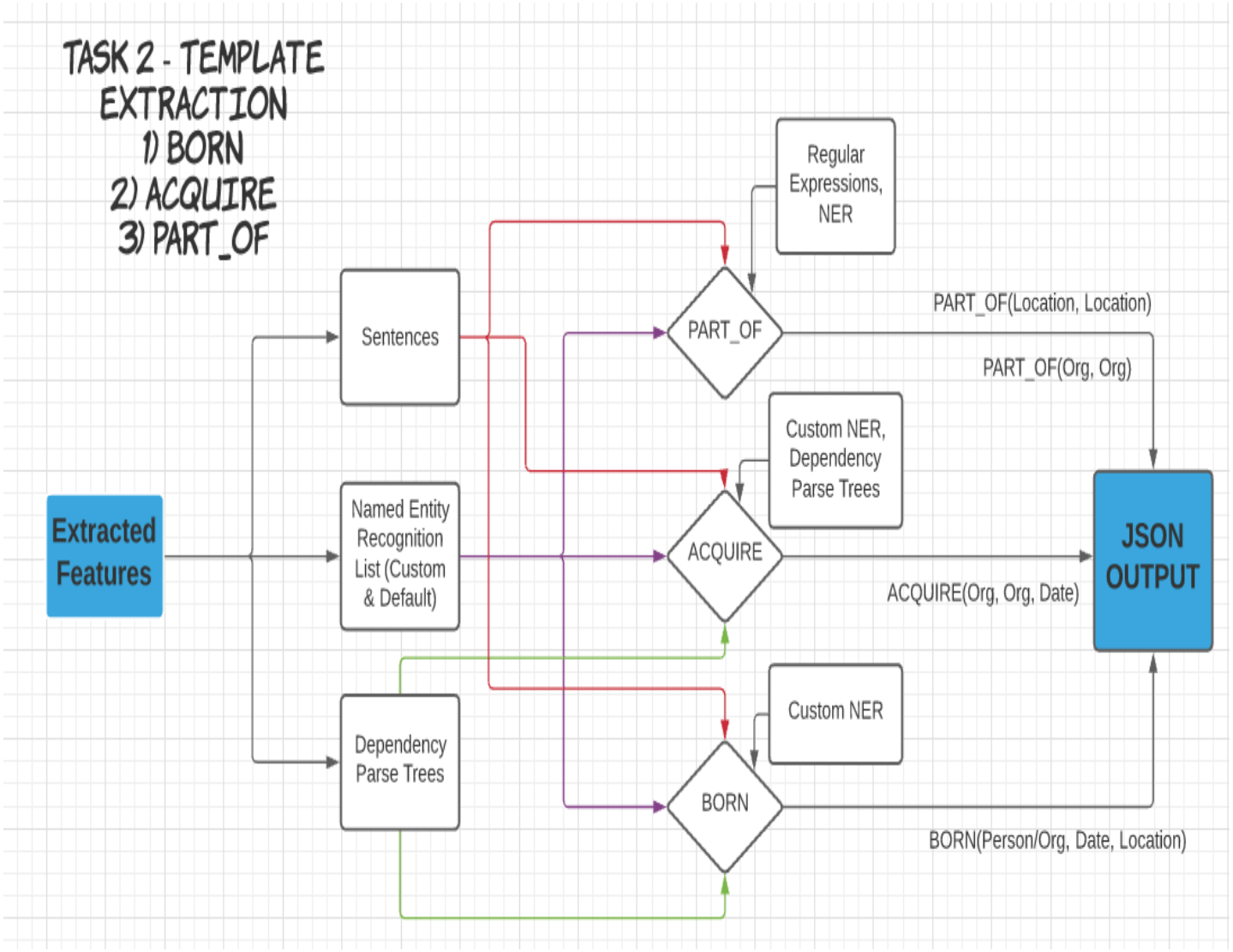
We understood that to build a successful Information Extraction application, we must first understand the input text. We will be using different NLP packages and libraries such as NLTK and spaCy to obtain necessary patterns and information from these input documents. For template extraction, we first need to extract features from input documents that could be utilized to identify the patterns. These features include Part-Of-Speech tags, dependency parse trees, WordNet features and more. Using these features, we created a rule-based heuristic approach model to extract the desired templates.

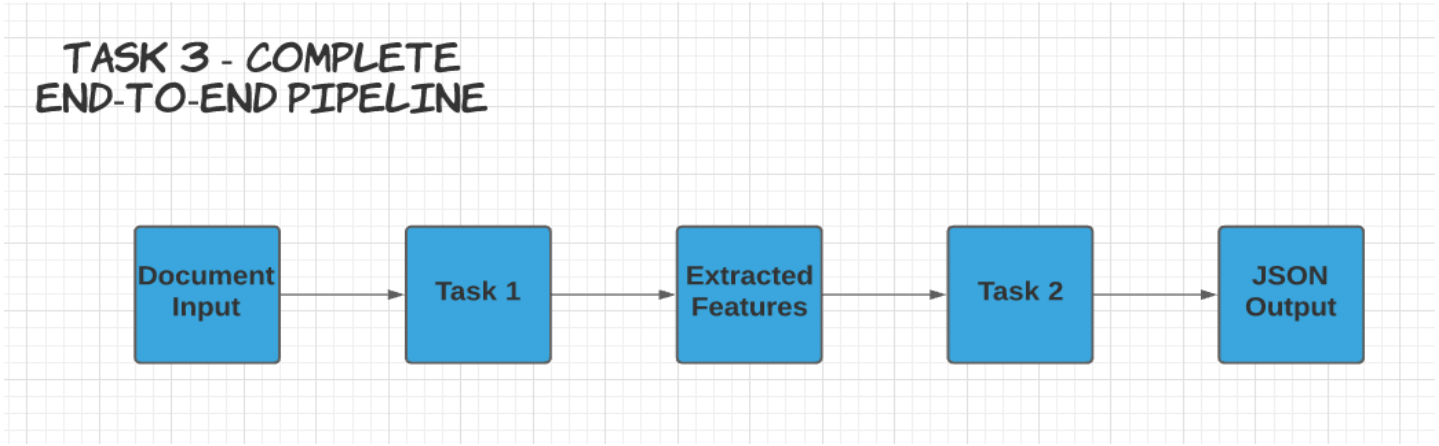
Implementation Details

Tools Used	*Description*
Python 3.7.9 - please ensure that you are using this version or earlier versions than this as neuralcoref only works with Python 3.7 and below	A high-level programming and scripting language utilized in projects of different scope and scale. It is extremely popular in NLP space due to its simple and understandable syntax, transparent semantics, and ever-expanding text processing tools.
Nltk 3.6.2	A leading platform/suite of libraries and tools for building Python programs to work with human language data, providing access to over 50 corpora/lexical resources (i.e. WordNet)
Spacy 2.1.0	A highly popular library for NLP that features NER, POS tagging, dependency parsing, word vectors and more.
En_core_web_sm-2.1.0	A pre trained English pipeline trained on written web text (blogs, news, comments), that includes vocabulary, vectors, syntax and entities.
Neuralcoref 4.0	A coreference resolution module is based on the super-fast spaCy parser and uses the neural net scoring model

Architectural Diagram




TASK 2 - TEMPLATE EXTRACTION
1) BORN
2) ACQUIRE
3) PART_OF
Regular Expressions, NER
Sentences
PART_OF
PART_OF(Location, Location)
PART_OF(Org, Org)
Custom NER, Dependency Parse Trees
Extracted Features
Named Entity Recognition List (Custom & Default)
ACQUIRE
JSON OUTPUT
ACQUIRE(Org, Org, Date)
Dependency Parse Trees
Custom NER
BORN
BORN(Person/Org, Date, Location)

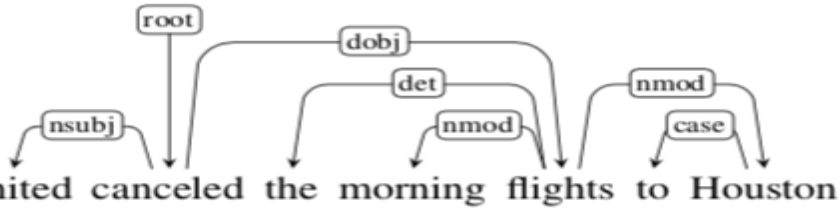

TASK 3 - COMPLETE END-TO-END PIPELINE
Document Input
Task 1
Extracted Features
Task 2
JSON Output

Task Descriptions

In order to meet the goal of this project as well as build an application that can be expanded upon, we designed our project with the principles of functional programming. This helps with the maintainability of our code. Below, we are detailing the purpose and implementation method for each of the tasks given.

Task 1 - Deep NLP Pipeline & Feature Extraction

- Segmentation -> Document to Sentences
  - Function: sentence_tokenizer(text)
  - The purpose of this function is to convert the text in the document into individual sentences.
- Word Tokenization -> Sentences to Words
  - Function: word_tokenizer(sentence)
  - The purpose of this function is to convert sentences to words. We are using the NLTK library to accomplish this functionality.
- Lemmatization
  - Function: lemmatization(word_tokens)
  - The purpose of this function is that it converts words from the sentences to its basic/root form(lemmas). We are using the NLTK library to accomplish this functionality.
- POS_Tagging
  - Function: pos_taggers(word_tokens)
  - The purpose of this function is to assign the Part-Of-Speech tags to the word tokens obtained earlier. We are using the NLTK library to accomplish this functionality.
- Hypernyms, Hyponyms, Meronyms, Holonyms, and Synonyms
  - Function: wordnet_features(word_tokens)
  - The purpose of this function is to extract the above semantic relations for the word_tokens. We are using the WordNet extension package from the NLTK library to accomplish this functionality.
- Dependency Parsing
  - Function: dependency_parsing(sentence)
  - The purpose of this function is to find the word dependency tags in a sentence. Our output will be a parse tree for the sentence. We utilized spaCy in-built dependency parser to accomplish this functionality.
  - Example of how it works is shown below:



- In the above sentence, the “head” is canceled and the dependent of that head is “flights”. This Head-Dependent relationship us recognize the semantic relationships between the words in our sentences

- Named Entity Recognition (NER)
  - Function: named_entity_recogniztion(sentence)
  - The purpose of this function is to find the default and custom entities in a sentence. These entities are labels for particular patterns in the sentence. We utilized spaCy’s in-built NER and customized it to include specific patterns for our task at hand.
  - Example of how spaCy’s NER works is shown below:

```
import spacy

nlp = spacy.load('en_core_web_sm')

sentence = "Apple is looking at buying U.K. startup for $1 billion"

doc = nlp(sentence)

for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
```

    - In the example above, the output will specify the following. The first column will be the entity, the next 2 columns will be the start/end characters in the sentence, and one last column defining category.

- Coreference Resolution
  - We utilized the neuralcoref model to perform pronoun resolution. We added this to our NLP pipeline.
  - Generic Concept Example of Coreference Resolution is shown below:
    - Sentence: *“Sam saw a red star travel for a while. The beauty of it mesmerized his heart, as he looked up out of the patio, and his aspiration returned to him.*
    - We first identify the potential spans in the sentence such as “Sam”, ‘a red star”, “it”, “his”, “he”. Then we need to combine the spans into groups with labels. The last step is to replace the pronouns - “it”, “his”, “he” - with the real-world entities in the sentence. The goal of this technique is to find, group and substitute ambiguous expressions in sentences to their real-world entities.

T*ask 2 - Template Extraction*

- BORN
    - Input: Sentence
    - Output: BORN(Person/Organization, Date, Location)
    - Implementation Description
        - We have defined custom NERs for this template extraction. To define these NERs, we created a custom entity named “BORN” and looked for patterns such as: “founded by”, “founded on”, “born” and few more. From there, based on the NER lists we acquired in Task 1 for each sentence, we filled the template by looking for organization(ORG) or PERSON tags for the first parameter, a DATE entity tag for the second parameter, and a location(GPE) tag for the third parameter.
- ACQUIRE
    - Input: Sentence
    - Output: ACQUIRE(Organization, Organization, Date)
    - Implementation Description
        - We have defined custom NERs for this template extraction. To define these NERs, we created a custom entity named “ACQUIRE” and looked for patterns such as: “acquired by”, “acquired”, “acquires”. Along with the custom NER, we utilized outputted dependency parse tree from Task 1 to find relations between the words in the sentence.
- PART_OF
    - Input: Sentence
    - Output
        - PART_OF(Organization, Organization)
        - PART_OF(Location, Location)
    - Implementation Description
        - We have used the default NERs for the sentences and looked for the location entity using regular expressions/patterns. For example, one of the types of patterns we utilized is (city, state). For the Part_Of with organization template, we are using the default NERs to find ORG entities and regular expressions to search for “Part of” relations in the sentences.

Task 3 - Information Extraction w/ Previous Tasks

- This task encompasses both previous tasks to run this project as an end-to-end application
- We accomplished this in 2 different ways:

- The project takes 1 input file and output the json file with the required features and 3 templates, OR
- The project performs this task where the input is a folder with multiple text documents.

Results/Error Analysis

Below are a few screenshots of our JSON output results extracted in the desired template format.


```
{
  "template": "ACQUIRE",
  "sentences": [
    "Later that month, Twitch\n\nTwitch was acquired by Amazon for $970 million."
  ],
  "arguments": {
    "1": "Amazon",
    "2": "Twitch",
    "3": "month"
  }
},
```

```
{
  "template": "BORN",
  "sentences": [
    "Amazon was founded by Jeff Bezos in Bellevue, Washington, in July 1994."
  ],
  "arguments": {
    "1": "Amazon",
    "2": "July 1994",
    "3": "Bellevue, Washington"
  }
},
```

```
    {
      "template": "PART_OF",
      "sentences": [
        "Amazon was founded by Jeff Bezos in Bellevue, Washington, in July 1994."
      ],
      "arguments": {
        "1": "Bellevue",
        "2": "Washington"
      }
    },
```

```
{
  "template": "ACQUIRE",
  "sentences": [
    "On February 14, 2019, Apple Inc. acquired DataTiger for Apple Inc. digital marketing technology."
  ],
  "arguments": {
    "1": "Inc.",
    "2": "technology",
    "3": "2019"
  }
},
```

Issues/Problems Encountered

1) Named Entities for ACQUIRE and BORN were not a part of the default NER setup in spaCy, so we had to customize the NER to include patterns for these 2 templates.
2) While browsing through the text documents, we found that there was some text which were formatted differently i.e., some text would not be left aligned as the rest of the text in the document. To overcome this, the sentence tokenizer was modified to accommodate these variations in input text documents.

Pending Issues

1) spaCy's in-built NER incorrectly classifies some person entities as organization and vice-versa.
2) Sometimes, the whole compound terms cannot be identified through the current level of the dependency parse trees.
3) The coreference resolution sometimes gives incorrect resolution for certain contexts.

Potential Improvements

1) To overcome the in-built NER incorrect classification issue, custom NER models can be created in order to rectify the classification (thereby increasing the accuracy).
2) We can build a deep learning model with annotated data to increase the accuracy of the information template extraction.
3) Better, optimized rules can yield better results for the templates.