CS 6320 – Natural Language Processing

Spring 2020

Project Report

Team - Alpha

Information Extraction

Dileep Gangadhar Godithi – dxg180045

Meghana Solleti – mxs180149

Project Description:

Implement an Information extraction application using NLP features and techniques. Information extraction is the task of extracting information from the unstructured documents. It automatically extracts information such as entities, relation between entities, etc.,

In this project the main task of information extraction is template filing. Template filling means extracting the required arguments from the sentences of the given documents.

Examples of Templates:

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

Sentence: 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”, “”, “”)

WORK Template: WORK (Person, Organization, Position, Location)

Sentence: Steven Paul Jobs was the chairman, chief executive officer (CEO), and co-founder of Apple Inc.

Extracted Template: WORK (“Steven Paul Jobs”, “Apple Inc.”, “chairman ; chief executive officer (CEO); co-founder”, “”)

PART Template: PART (Location, Location)

Sentence: Richardson is a principal city in Dallas and Collin counties in the U.S. state of Texas.

PART (“Richardson”, “Dallas”)

Proposed Solution:

Buy Template:

Implemented by creating certain rules and heuristics based on dependency parse on individual sentences. Basic approach is to find the root of the dependency parser to find a template match as root is generally a verb. If the root is one of lemmas of buy, acquire or purchase then the sentence is a match for the template and additional rules and branches in the parser are explored.
Sentences in active voice follows ’nsubj’🡐root🡒 ‘dobj’ structure and sentences in passive voice have ‘nsubjpass’🡐root structure.

Additional arguments like amount, quantity and source are extracted from the right children of root and their subtrees. Followed DFS based approach to parse deeper into the subtrees.
Exceptions are added based on the initial result which provided better generalization. Using the Named Entities from Spacy and prepositional phrase structures arguments are found.

Work Template:

Implemented a heuristic approach using NLP features to fill the extracted information in the template. As this is work template we are dealing with the following entities person, organization, job-title, location.

Here we incorporated the entities derived from Stanford Core Nlp into Spacy pipeline by adding the custom entity. Because spacy was not recognizing the job-titles. In this template we have to identity a person working in an organization with x job-title and at y location.

First get the sentences of the given document. The sentence is now passed through the pipeline of spacy library. We need to get the named entities of the sentence. The main idea here is to find out relation between entities based on rules written using dependency parsing and named entity recognition. Parsing the tree to get entity type relevant to arguments of template and finding paths between them to find a relation.

Additional rules are written for pattern matching where sentence structure is not found if named entities are similar to tabular form.

<u>Part Template</u>:

Implemented a heuristic approach using NLP features to fill the extracted information in the template. As this is part template we are dealing with the locations. We have to find out locations (x , y) such that x is part of y.

First get the sentences of the given document. The sentence is now passed through the pipeline of spacy library. We need to get the named entities of the sentence and get a collection of locations. The main idea here is to find out relation between locations based on rules written using dependency parsing and named entities.

We iterate over the sentence and find out the edges using head and children relation in the dependency parser. If (A,B) is edge then A is a token in dependency parse tree whose child is B. Using these edges we can create directed and undirected graph.

Now we got the collection of locations in the sentence and also the graphs which represents the dependencies of the tokens of sentence. For all combinations of locations we have to find the correct relations based on the rules written using graph. Two locations a, b is said to be in part relation if they have directed path or an undirected path, with some specific prepositions like (is , in ,of, among, are) joining these locations in the path.

Explanation with example:

Sentence: Richardson is a principal city in Dallas and Collin counties in the U.S. state of Texas.

Locations in Sentence: ['Richardson', 'Dallas', 'U.S.', 'Texas']

Edges in the sentence: [('is', 'richardson'), ('is', 'city'), ('is', '.'), ('city', 'a'), ('city', 'principal'), ('city', 'in'), ('city', 'in'), ('in', 'counties'), ('dallas', 'and'), ('dallas', 'collin'), ('counties', 'dallas'), ('in', 'state '), ('state', 'the'), ('state', 'u.s.'), ('state', 'of'), ('of', 'texas')]

Path from richardson to dallas : ['richardson', 'is', 'city', 'in', 'counties', 'dallas']

As there is path between richardson to dallas and also connected with the prepositions ‘is’, ‘in’ so we can conclude that richardson is part of dallas.

Template extracted: {'template': 'PART', 'sentences': ['Richardson is a principal city in Dallas and Collin counties in the U.S. state of Texas.'], 'arguments': {'1': 'richardson', '2': 'dallas'}}

IMPLEMENTATION DETAILS

Programming Tools:

- Language - Python

  Spacy

  NLTK

  Stanford NLP

  Networkx

  Neuralcoref

- IDE – Jupyter/Spyder
- http://corenlp.run/
- https://explosion.ai/demos/displacy
- https://explosion.ai/demos/displacy-ent

Results and Error Analysis:

BUY-TEMPLATE:

```
1.  "template": "BUY",
   "sentences": [
     "In 2017, Amazon acquired Whole Foods Market for US$13.4 billion, which vastly
increased Amazon's presence as a brick-and-mortar retailer."
   ],
   "arguments": {
     "1": "Amazon",
     "2": "Whole Foods Market",
     "3": "US$13.4 billion",
     "4": "",
     "5": ""
   }

2.  {
   "template": "BUY",
   "sentences": [
     "In 1945, businessman and former J. C. Penney employee Sam Walton bought a branch
of the Ben Franklin stores from the Butler Brothers."
```

```
        ],
        "arguments": {
          "1": "businessman",
          "2": "a branch",
          "3": "",
          "4": "",
          "5": "the Butler Brothers"
        }
```

Extraction 1 is accurate but extraction 2 have additional issues due to syntactic ambiguities as sometimes preposition ‘of’ can refer to argument-source

PART-TEMPLATE:
Richardson is a principal city in Dallas and Collin counties in the U.S. state of Texas.

```
{
'template': 'PART',
'sentences': ['Richardson is a principal city in Dallas and Collin counties in the U.S. state of Texas
.'], '
arguments': {
'1': 'u.s.',
'2': 'texas'
}
}
```

Some part relations may not be accurate, which can be resolved incorporating more rules.

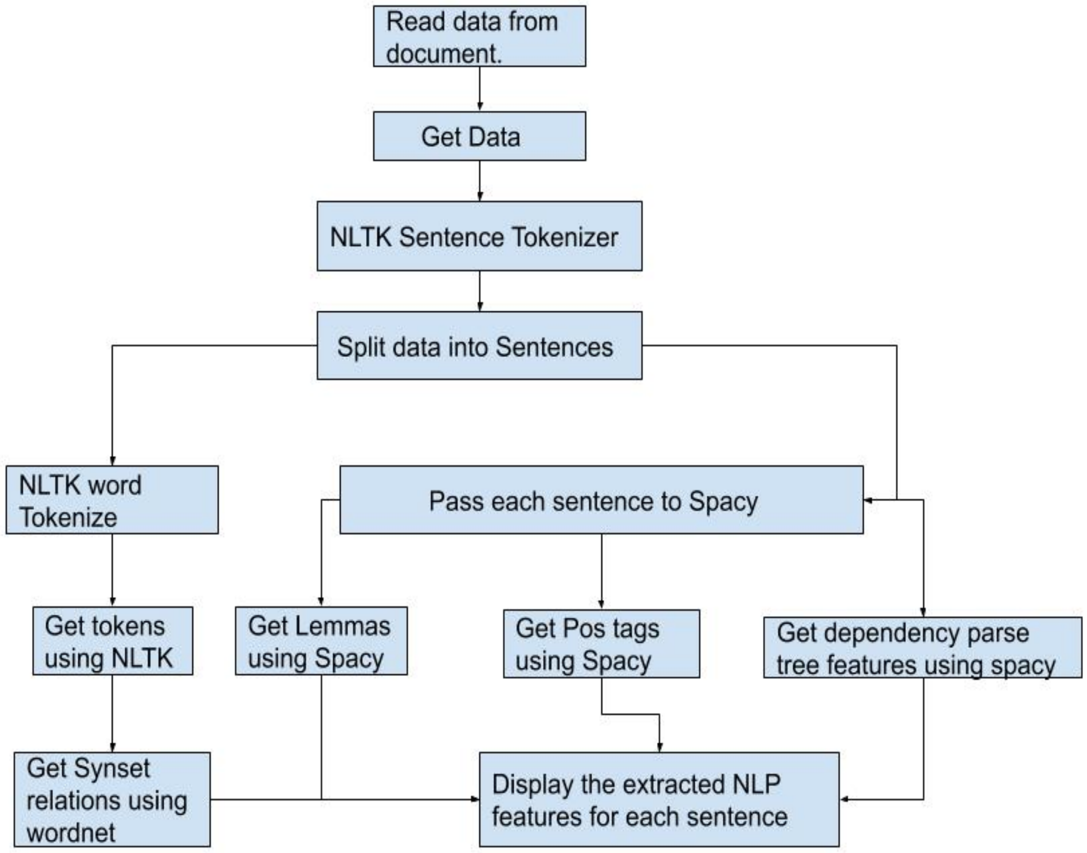
WORK-TEMPLATE:

```
{
        "template": "WORK",
        "sentences": [
          "John Furner has been the CEO of Sam's Club since early 2017.\n"
        ],
        "arguments": {
          "1": "Furner",
          "2": "Club",
          "3": "CEO",
          "4": ""
        }
}
```

Due to real world issues in resolving proper nouns as perfect named entities, only part of them are generated in template

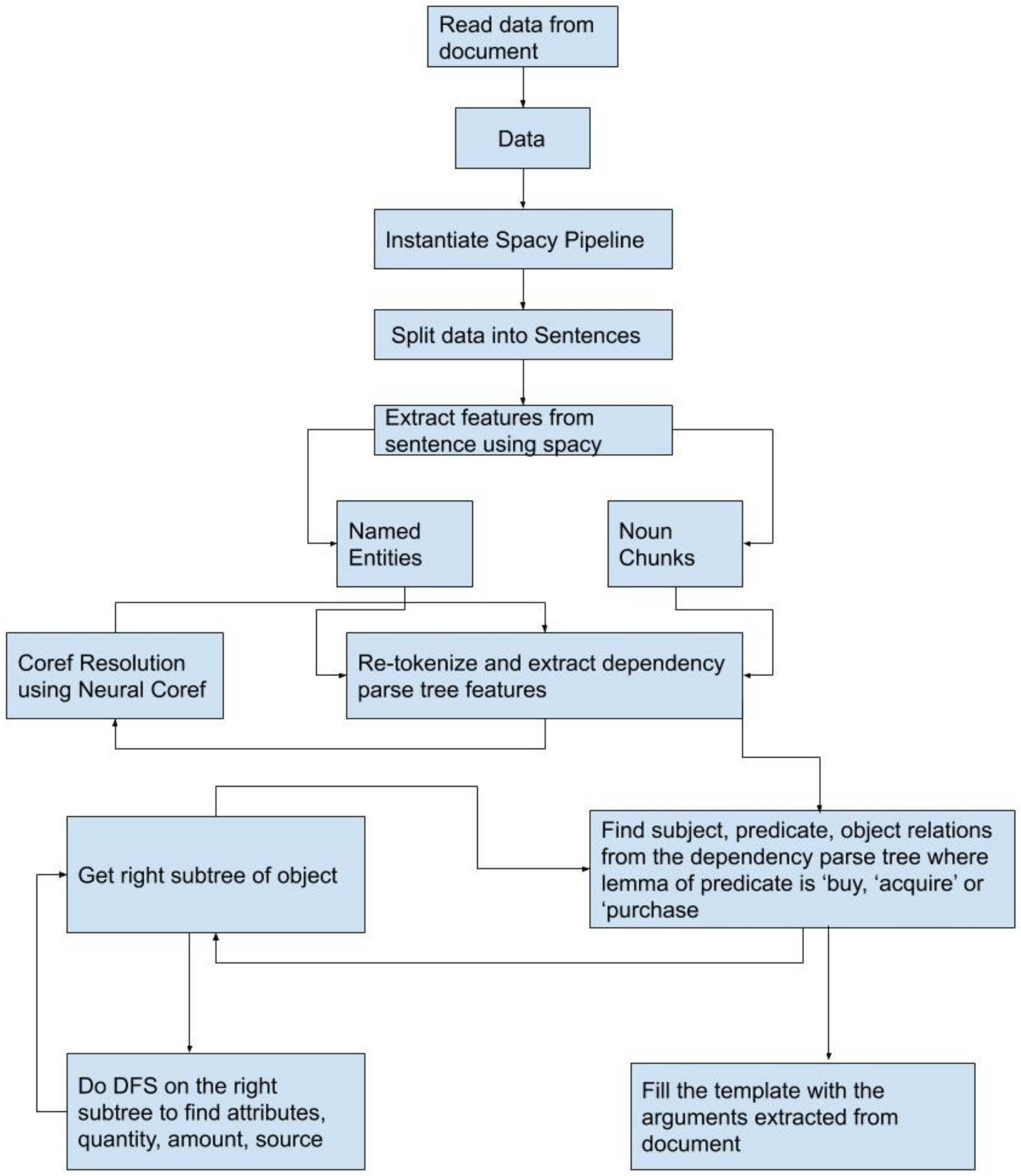
Architectural Diagram:

Task1:

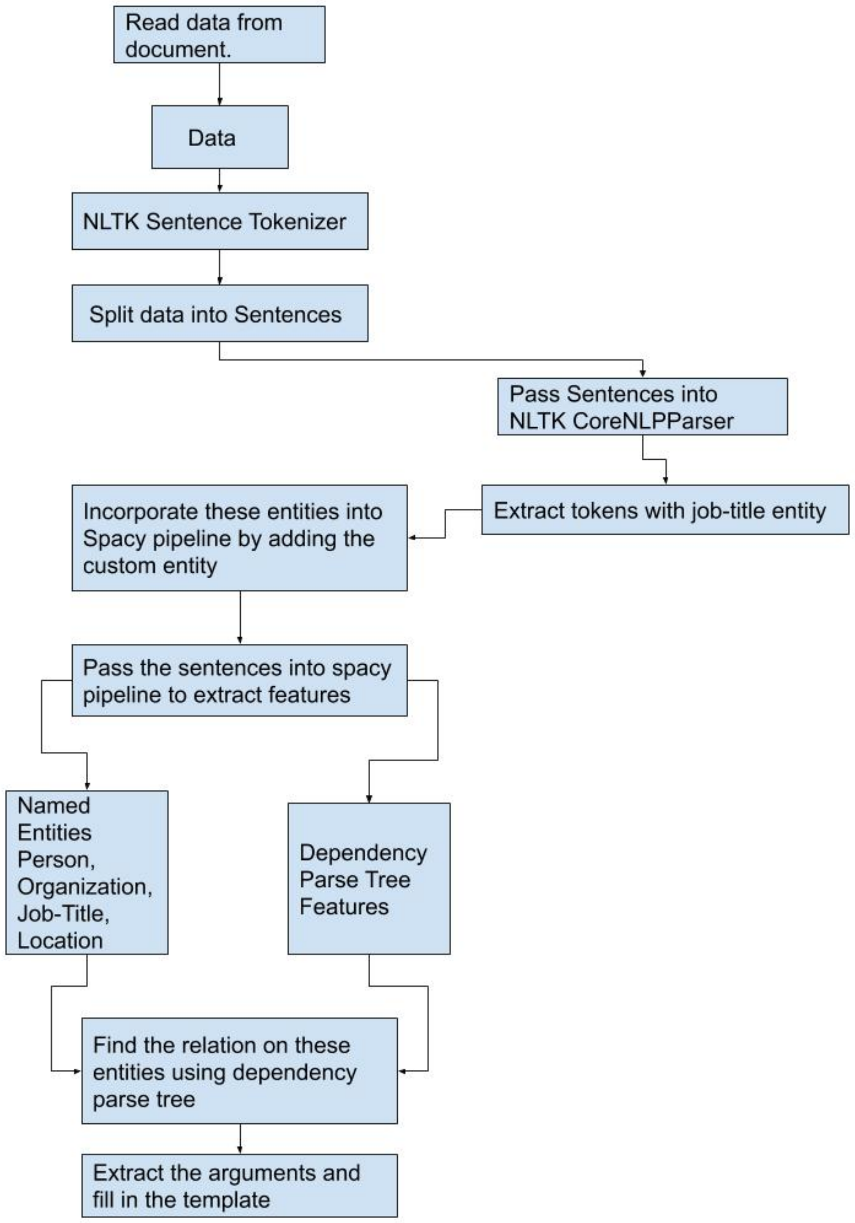

Read data from document.
Get Data
NLTK Sentence Tokenizer
Split data into Sentences
NLTK word Tokenize
Pass each sentence to Spacy
Get tokens using NLTK
Get Lemmas using Spacy
Get Pos tags using Spacy
Get dependency parse tree features using spacy
Get Synset relations using wordnet
Display the extracted NLP features for each sentence

Task2:

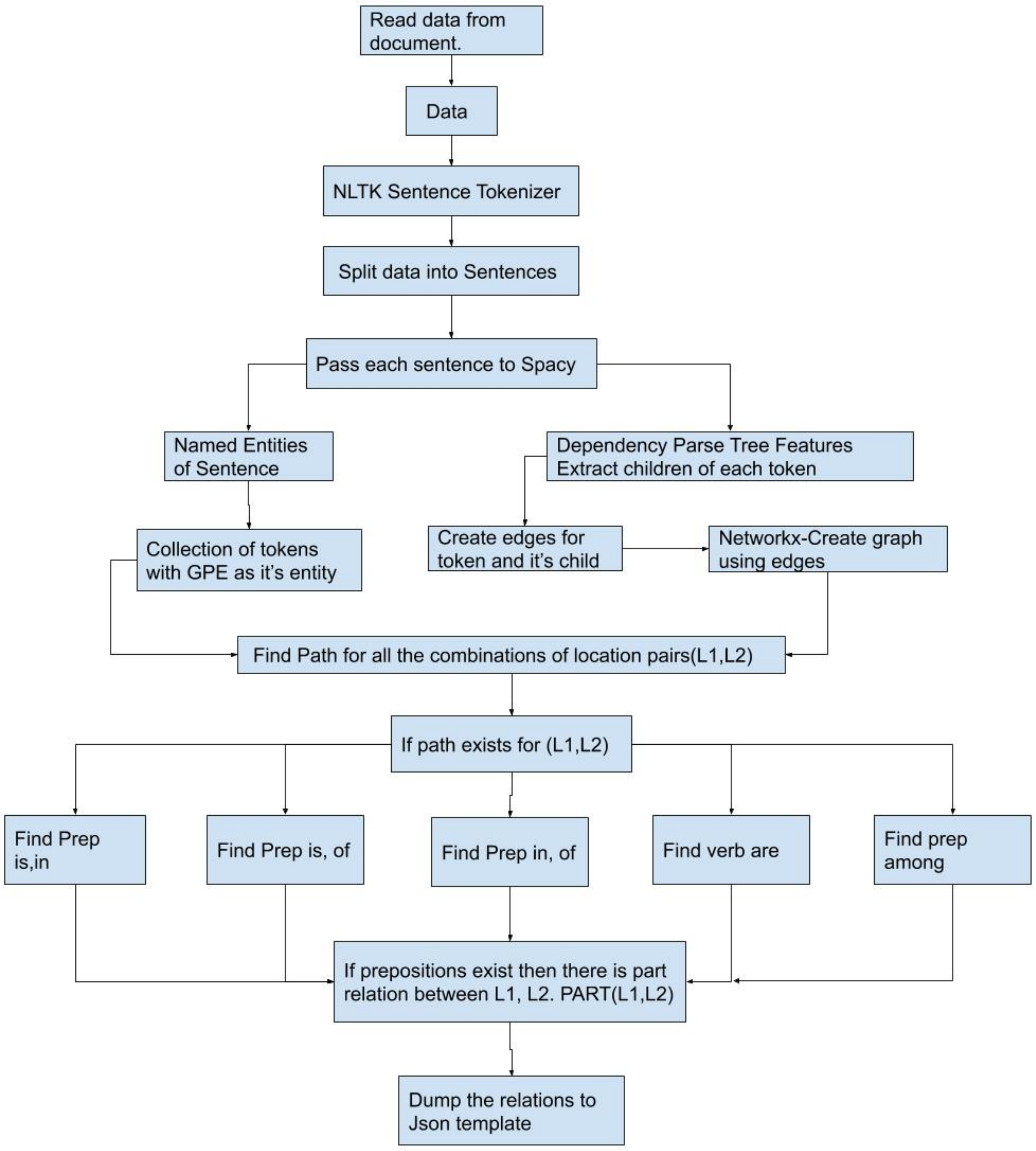
BUY-TEMPLATE Architectural Diagram


Read data from document
Data
Instantiate Spacy Pipeline
Split data into Sentences
Extract features from sentence using spacy
Named Entities
Noun Chunks
Coref Resolution using Neural Coref
Re-tokenize and extract dependency parse tree features
Get right subtree of object
Find subject, predicate, object relations from the dependency parse tree where lemma of predicate is 'buy, 'acquire' or 'purchase
Do DFS on the right subtree to find attributes, quantity, amount, source
Fill the template with the arguments extracted from document

WORK-TEMPLATE Architectural Diagram:


Read data from document.
Data
NLTK Sentence Tokenizer
Split data into Sentences
Pass Sentences into NLTK CoreNLPParser
Extract tokens with job-title entity
Incorporate these entities into Spacy pipeline by adding the custom entity
Pass the sentences into spacy pipeline to extract features
Named Entities Person, Organization, Job-Title, Location
Dependency Parse Tree Features
Find the relation on these entities using dependency parse tree
Extract the arguments and fill in the template

PART-TEMPLATE Architectural Diagram


Read data from document.
Data
NLTK Sentence Tokenizer
Split data into Sentences
Pass each sentence to Spacy
Named Entities of Sentence
Dependency Parse Tree Features Extract children of each token
Collection of tokens with GPE as it's entity
Create edges for token and it's child
Networkx-Create graph using edges
Find Path for all the combinations of location pairs(L1,L2)
If path exists for (L1,L2)
Find Prep is,in
Find Prep is, of
Find Prep in, of
Find verb are
Find prep among
If prepositions exist then there is part relation between L1, L2. PART(L1,L2)
Dump the relations to Json template

Problems Encountered:

Some of the problems encountered while extracting the information using NLP features from spacy, nltk.

Problem: Spacy recognized some words with wrong entities, for example Richardson was recognized as person instead of location which caused ambiguity.
Solution: Identified such type of words and included them in the spacy pipeline as customized entities.

Problem: Spacy was unable to recognize all the named entities such as Job title. We can't extract job titles using spacy.
Solution: Extracted the job titles using Stanford Core Nlp and incorporated these entities into spacy as customized entities.

Problem: Coreference resolution package have binaries incompatibility issue with Spacy for latest version of packages.
Solution: Resolved by using workaround to downgrade Spacy library

Pending Issues:

Resolving the argument 'source' in buy template is incorrect sometimes due to ambiguity on prepositional phrase.

Multiple buy/acquire matches in single sentences are not resolved every time.

Potential Improvements:

Better coreference resolution should be found as depending on existing packages are introducing additional issues

Acquiring already annotated data or perform annotations to use deep learning or machine learning techniques to improve results.

Incorporating better relation extraction techniques and forming a knowledge graph.

Summary: Implemented an Information extraction technique to fill the templates using various features of NLTK, Spacy, StanfordCoreNlp. Used all these libraries to extract syntactic and semantic information. Extracted named entities, dependency parse tree features, lemmas, synsets and incorporated all these features to design an information extraction technique.

References:

https://spacy.io/usage/rule-based-matching
https://spacy.io/usage/examples
https://www.nltk.org/book/ch07.html#ex-ie4
https://towardsdatascience.com/natural-language-processing-using-stanfords-corenlp-d9e64c1e1024
https://www.analyticsvidhya.com/blog/2019/09/introduction-information-extraction-python-spacy/