Information Extraction, InfEx2020

Ananya Banerjee

The University of Texas at Dallas
Richardson, TX 75080 USA
{axb170053@utdallas.edu}

1

Abstract. *Information extraction from natural language forms such as text and speech, is one of the most challenging tasks known to academia. There have been various different approaches towards reasonable and efficient extraction of information. These approaches can be broadly classified into Rule Based, Supervised, Distantly Supervised, Semi-Supervised and Unsupervised. Instead of using one approach to extract information, our proposed methodology believes in utilization of the best of all of these approaches and devises a unique paralleled hybrid approach to information extraction.*

Keywords*: Natural Language Processing, Relationship Extraction, Coreference Resolution.*

1. Introduction

Extraction of information is a particularly daunting task in the present world specially due to the humongous amounts of data that we collect everyday. Our approach in this project focuses on utilization of the syntactic, semantic and lexical information of the data (text) given in the documents. We use a collection of Wikipedia articles to help facilitate our research.

2. Problem Definition

Our problem definition was divided into three main tasks. In the first task, we were required to implement a deep Natural Language Processing pipeline to extract the following NLP based features from the text articles/documents:

- Split the document into sentences
- Tokenize the sentences into 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 get parse-tree based patterns as features

- Using WordNet, extract hypernymns, hyponyms, meronyms, AND holonyms as features

The second task was to implement a machine-learning, statistical, or heuristic (or a combination) based approach to extract filled information templates from the corpus of text articles.

The template for each relationship extraction task is as shown below:

- Template #1:

  BUY(Buyer, Item, Price, Quantity, Source)
- Template #2:

  WORK(Person, Organization, Position, Location)
- Template #3:

  PART(Location, Location)

The third task was to create a Information Extraction application which incorporates the above two tasks and takes a Wikipedia articles as a document and extract information about three major kinds of relationships, namely, BUY, WORK and PART (as shown above) and store them in a json file of the format as shown below.

```
{
        "document": "AppleInc.txt",
    "extraction": [
                {
                        "template": "WORK",
                        "sentences": ["Steven Paul Jobs (; February 24, 1955 – October 5, 2011) was an American business magnate and investor.", "He was the
chairman, chief executive officer (CEO), and co-founder of Apple Inc.; chairman and majority shareholder of Pixar; a member of The Walt Disney Company's board
of directors following its acquisition of Pixar; and the founder, chairman, and CEO of NeXT.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": "Steven Paul Jobs",
                                "2": "Apple Inc.",
                                "3": "chairman ; chief executive officer (CEO); co-founder",
                                "4": ""
                        }
                },
```

3. Text Preprocessing and Feature Extraction

We did the following to pre-process our given document and prepare the features needed to be used later on.

- Split the document into sentences
- Perform dependency parsing or full-syntactic parsing to get parse-tree based and save them.
- Part-of-speech Tagging (POS- Tagging)
- Perform Named Entity Recognition
- Tokenize the sentences into words
- Removing Stop words
- Remove Special Characters
- Lemmatize the words to extract lemmas as features

- Remove html tags
- Using WordNet, extract hypernymns, hyponyms, meronyms, AND holonyms as features
- Perform Coreference Resolution

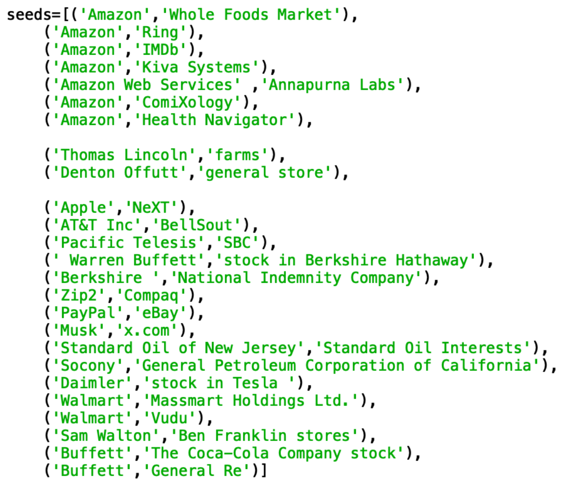
4. Proposed Solution

The solution we propose uses a combination of Semi-Supervised, Rule Based and Self-devised techniques to identify the three main relationships : BUY, WORK and PART.

Let's talk about each relationship one by one.

For the relationship **BUY**, our solution uses a three major techniques:

- **Technique 1 (Semi-Supervised):**
  - Here we take few seed tuples of the form (M1, M2) as input.
  - Then we find sentences in documents containing those seeds.
  - The next step is to establish a generalized context vector using Glove embedding. We consider three types of context vectors: Context before mention M1, context after Mention M2 and context between Mentions M1 and M2.
  - Once we get these context vectors, we try to extract newer patterns from these.
  - We use a confidence function to check how good our newer extracted patterns are. If they are above a threshold of 0.4, we allow these patterns to stay. Otherwise, the extracted patterns are deleted.
  - Then we extract newer seed tuples and add them to our list of original seed tuples and we iterate.



```
seeds=[('Amazon','Whole Foods Market'),
    ('Amazon','Ring'),
    ('Amazon','IMDb'),
    ('Amazon','Kiva Systems'),
    ('Amazon Web Services' ,'Annapurna Labs'),
    ('Amazon','ComiXology'),
    ('Amazon','Health Navigator'),

    ('Thomas Lincoln','farms'),
    ('Denton Offutt','general store'),

    ('Apple','NeXT'),
    ('AT&T Inc','BellSout'),
    ('Pacific Telesis','SBC'),
    (' Warren Buffett','stock in Berkshire Hathaway'),
    ('Berkshire ','National Indemnity Company'),
    ('Zip2','Compaq'),
    ('PayPal','eBay'),
    ('Musk','x.com'),
    ('Standard Oil of New Jersey','Standard Oil Interests'),
    ('Socony','General Petroleum Corporation of California'),
    ('Daimler','stock in Tesla '),
    ('Walmart','Massmart Holdings Ltd.'),
    ('Walmart','Vudu'),
    ('Sam Walton','Ben Franklin stores'),
    ('Buffett','The Coca-Cola Company stock'),
    ('Buffett','General Re')]
```

Figura 1. Seed Tuples Used for BUY

- **Technique 2 (Rule Based):**
  - Here we take use the Dependency parse of one sentence at a time to extract our relationship.

- We have a list called lis=['bought','buys','buy','acquires','acquire','acquired'] in place already.
- We check if ROOT of the dependency parse of the sentence is a verb $\in$ lis or not.
- If it is the ROOT, then we check whether its children are nsubj and dobj. If we find both of them then we are done as Subject=nsubj and Object=dobj.
- Else, if the ROOT is not a verb, then we look for all words of lis in our sentence's dependency parse. If we find any word $\in$ lis in our sentence's parse structure, then we check if that word's children contains both nsubj and dobj or not. If it contains both then we are done as Subject=nsubj and Object=dobj.
- Else, we check if we have atleast a object (dobj) in the children of the word (node) in the dependency parse tree. If we do, then we further check if the part of speech tag of the ROOT is NN or NNP or NNS. It it is, then we declare ROOT as nsubj and break. If ROOT doesn't have these three POS Tags then we do nothing.
- If none of these conditions are true, then we move over to the next sentence.

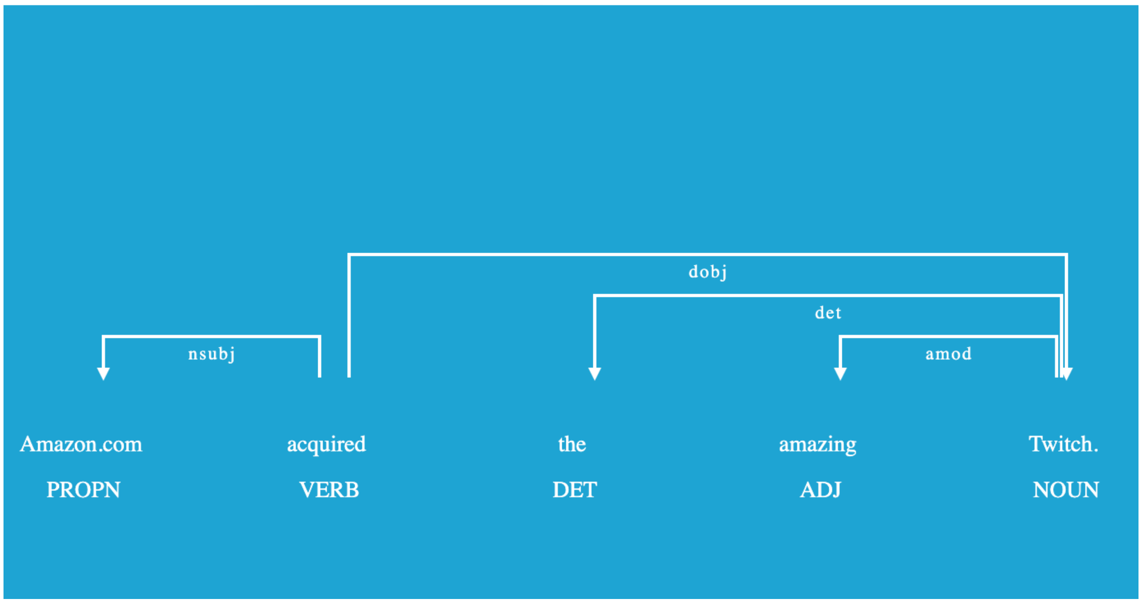
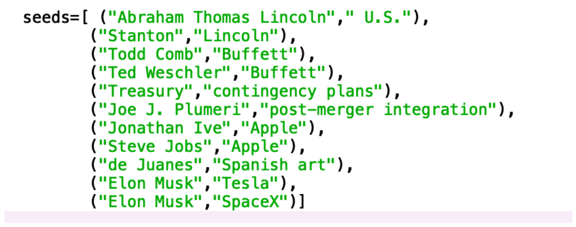


Figura 2. An example of Dependency Parse that we use to detect subject and object of predicate="acquired"

For the relationship **WORK**, our solution uses a three major techniques:

- **Technique 1 (Semi-Supervised):**
    - Here we take few seed tuples of the form (M1, M2) as input.
    - Then we find sentences in documents containing those seeds.
    - The next step is to establish a generalized context vector using Glove embedding. We consider three types of context vectors: Context before mention M1, context after Mention M2 and context between Mentions M1 and M2.
    - Once we get these context vectors, we try to extract newer patterns from these.
    - We use a confidence function to check how good our newer extracted patterns are. If they are above a threshold of 0.4, we allow these patterns to stay. Otherwise, the extracted patterns are deleted.
    - Then we extract newer seed tuples and add them to our list of original seed tuples and we iterate.



```
seeds=[ ("Abraham Thomas Lincoln"," U.S."),
        ("Stanton","Lincoln"),
        ("Todd Comb","Buffett"),
        ("Ted Weschler","Buffett"),
        ("Treasury","contingency plans"),
        ("Joe J. Plumeri","post-merger integration"),
        ("Jonathan Ive","Apple"),
        ("Steve Jobs","Apple"),
        ("de Juanes","Spanish art"),
        ("Elon Musk","Tesla"),
        ("Elon Musk","SpaceX")]
```

Figura 3. Seed Tuples Used for WORK

- **Technique 2 (Rule Based):**
    - Here we take use the Dependency parse of one sentence at a time to extract our relationship.
    - We have a list called lis=['work','works','worked','toil','labor'] in place already.
    - We check if ROOT of the dependency parse of the sentence is a verb $\in$ lis or not.
    - If it is the ROOT, then we check whether its children are nsubj and dobj. If we find both of them then we are done as Subject=nsubj and Object=dobj.
    - Else, if the ROOT is not a verb, then we look for all words of lis in our sentence's dependency parse. If we find any word $\in$ lis in our sentence's parse structure, then we check if that word's children contains both nsubj and dobj or not. If it contains both then we are done as Subject=nsubj and Object=dobj.
    - Else, we check if we have atleast a object (dobj) in the children of the word (node) in the dependency parse tree. If we do, then we further check if the part of speech tag of the ROOT is NN or NNP or NNS. It it is, then we declare ROOT as nsubj and break.
    - Else, we check if we have atleast a subject (nsubj) in the children of the word(node) in the dependency parse tree. If we do, then we look for pobj node in the children of node. The intuition behind pobj is that object of preposition is usually connected to a direct object (dobj). Thus, the children of pobj becomes our object and we break.

  - If none of these conditions are true, then we move over to the next sentence.

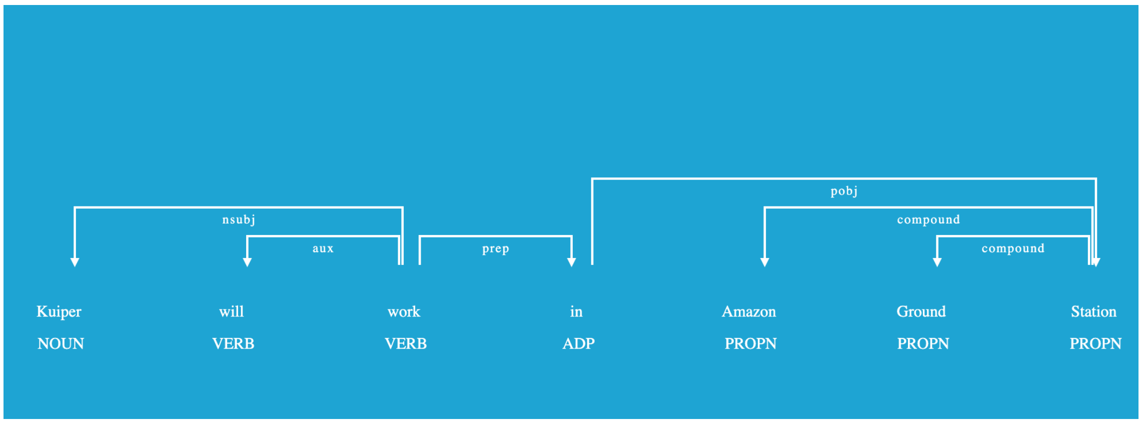


Figura 4. An example of Dependency Parse that we use to detect subject and object of predicate=”work”

For the relationship **PART**, our solution uses a three major techniques:

- **Technique 1 (Semi-Supervised):**
  - Here we take few seed tuples of the form (M1, M2) as input.
  - Then we find sentences in documents containing those seeds.
  - The next step is to establish a generalized context vector using Glove embedding. We consider three types of context vectors: Context before mention M1, context after Mention M2 and context between Mentions M1 and M2.
  - Once we get these context vectors, we try to extract newer patterns from these.
  - We use a confidence function to check how good our newer extracted patterns are. If they are above a threshold of 0.4, we allow these patterns to stay. Otherwise, the extracted patterns are deleted.
  - Then we extract newer seed tuples and add them to our list of original seed tuples and we iterate.

```
seeds=[("Kentucky","U.S."),
       ("Sinking Spring Farm","Hodgenville, Kentucky"),
       ("Hingham","sNorfolk"),
       ("Cupertino","California"),
       ("Nebraska","United States"),
       ("Norcross","Georgia"),
       ("China","East Asia"),
       ("Yellow River","North China Plain"),
       ("Himalayas","India"),
       ("Mount Everest"," Sino-Nepalese border"),
       ("Ayding Lake","Turpan Depression"),
       ("China","Asia"),
       ("Dallas","United States"),
       ("Dallas","North Texas"),
       ("Fort Worth","East Texas")]
```

Figura 5. Seed Tuples Used for PART

- **Technique 2 (Rule Based):**
  - Here we take use the Dependency parse of one sentence at a time to extract our relationship.
  - We have a list called lis=['situated','located'] in place already.
  - We check if ROOT of the dependency parse of the sentence is a verb $\in$ lis or not.
  - If it is the ROOT, then we check whether its children are nsubj and dobj. If we find both of them then we are done as Subject=nsubj and Object=dobj.
  - Else, if the ROOT is not a verb, then we look for all words of lis in our sentence's dependency parse. If we find any word $\in$ lis in our sentence's parse structure, then we check if that word's children contains both nsubj and dobj or not. If it contains both then we are done as Subject=nsubj and Object=dobj.
  - Else, we check if we have atleast a object (dobj) in the children of the word (node) in the dependency parse tree. If we do, then we further check if the part of speech tag of the ROOT is NN or NNP or NNS. It it is, then we declare ROOT as nsubj and break.
  - Else, we check if we have atleast a subject (nsubj) in the children of the word(node) in the dependency parse tree. If we do, then we look for pobj node in the children of node. The intuition behind pobj is that object of preposition is usually connected to a direct object (dobj). Thus, the children of pobj becomes our object and we break.
  - If none of these conditions are true, then we move over to the next sentence.

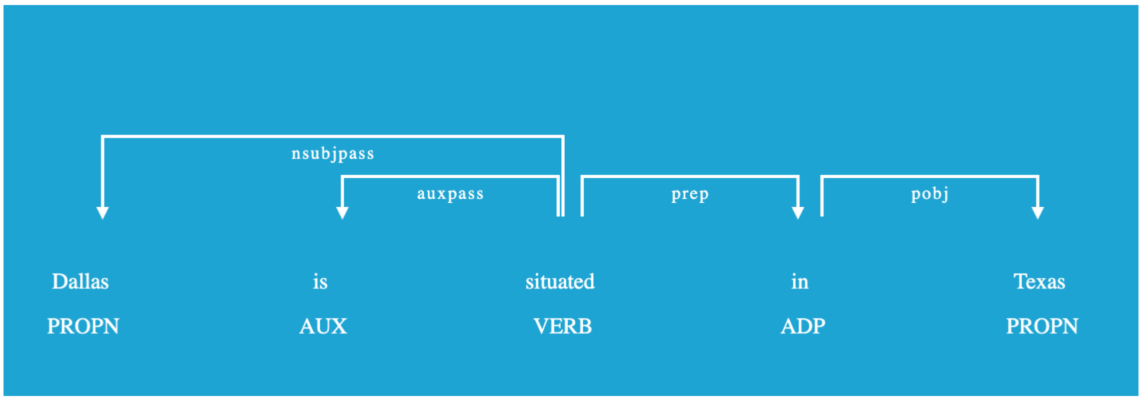


Figura 6. An example of Dependency Parse that we use to detect subject and object of predicate="situated"

- **Technique 3: Quirky Use of NER:**
  - We perform Named Entity Recognition on every line and collect all the entities.
  - If we have 2 geographical locations (GPE) as entities separated by comma (ENT 1, ENT2) then we call it PART relationship where ENT1 is a part of ENT2.
  - We further check if any of the detected entities are states in united states using a states.csb file and output that as a implied relationship too.

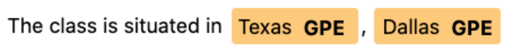


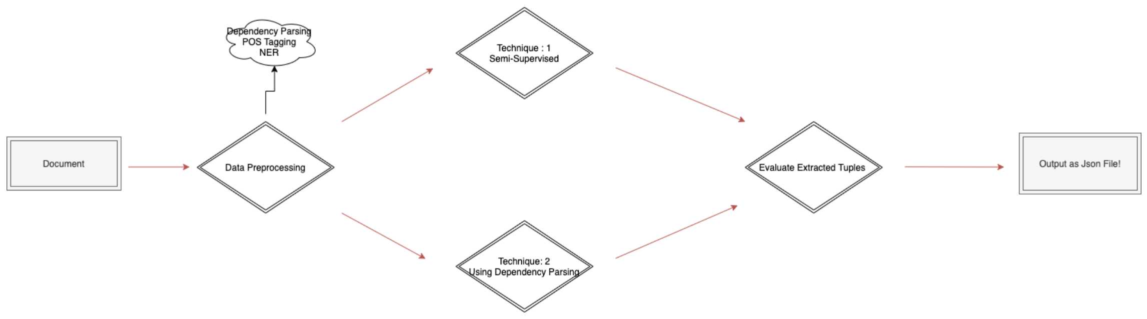
Figura 7. An example of NER used to detect subject and object of predicate="situated"

5. Methodology

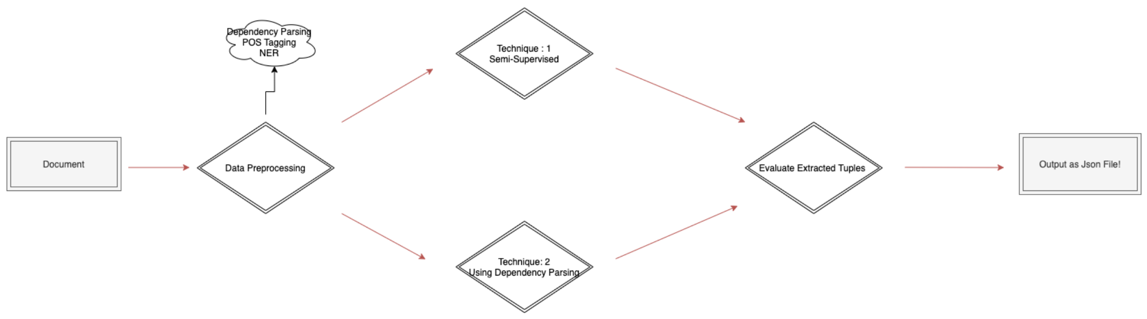
As far as implementation of the above proposed solution goes, it is pretty straight forward. The architectual diagrams given in sub section 5.1 will give you a concise idea about how the system works.

5.1. Architectural diagram

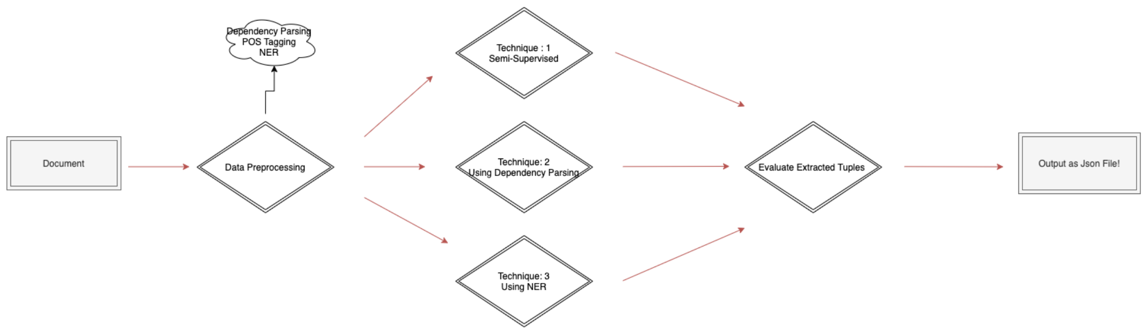
The architectural diagram of **BUY** is:



The architectural diagram of **WORK** is:



The architectural diagram of **PART** is:



5.2. Programming tools (including third party software tools used)

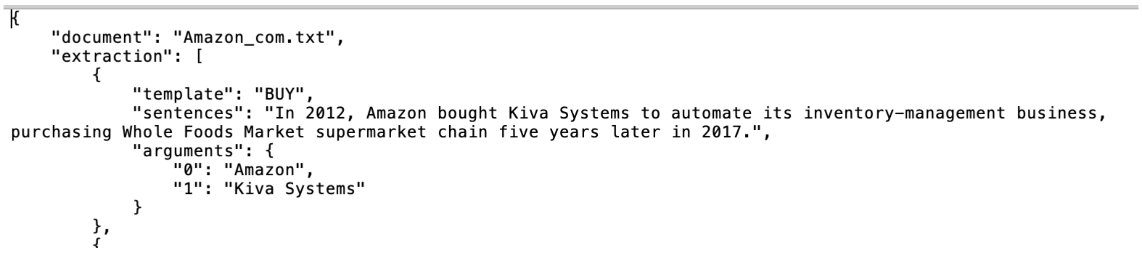
The list of programming libraries used are as follows:

- Spacy
- StanfordNLP
- Pandas
- Numpy
- NLTK
- Itertools
- PytextRank
- Re
- bs4 or BeautifulSoup
- Collections
- Geotext
- Json
- String
- Os
- Sklearn
- en_core_web_sm
- StanfordOpenIE
- Glove Embeddings

5.3. Results and Error analysis

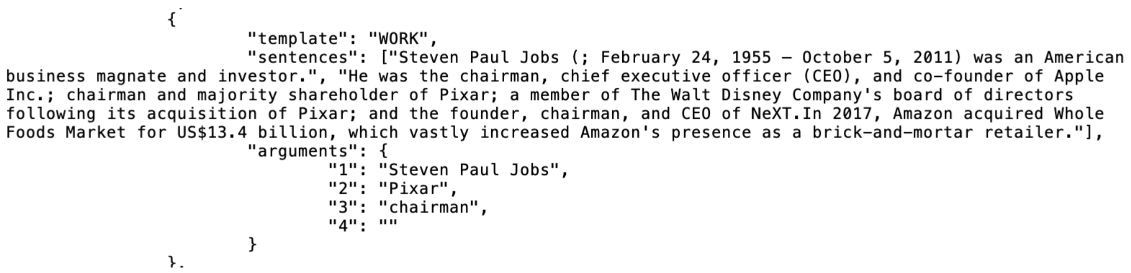
Now, let’s talk about the results! Some of the snapshots of the kind of results we get from our given documents are as shown below.

BUY



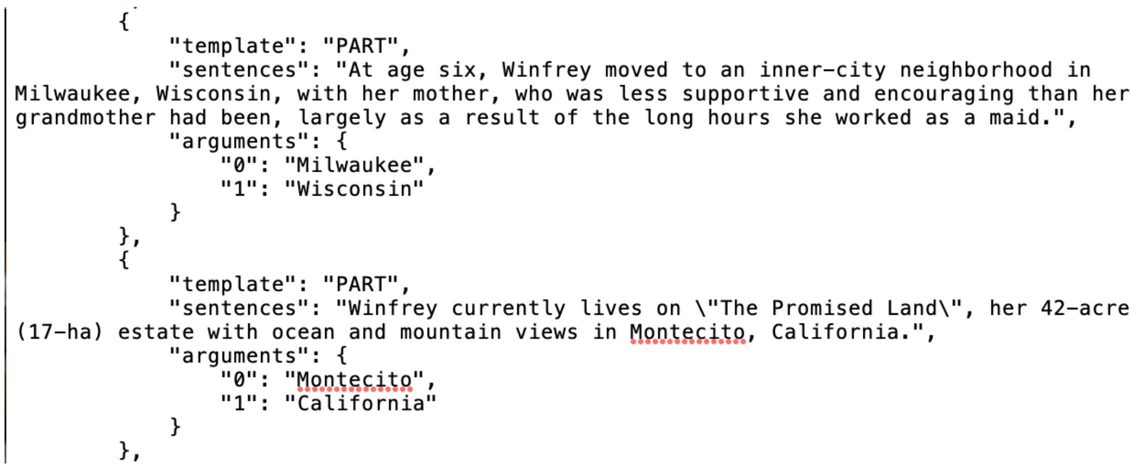
```
{
    "document": "Amazon_com.txt",
    "extraction": [
        {
            "template": "BUY",
            "sentences": "In 2012, Amazon bought Kiva Systems to automate its inventory-management business,
purchasing Whole Foods Market supermarket chain five years later in 2017.",
            "arguments": {
                "0": "Amazon",
                "1": "Kiva Systems"
            }
        },
        {
```

WORK



```
                {
                        "template": "WORK",
                        "sentences": ["Steven Paul Jobs (; February 24, 1955 – October 5, 2011) was an American
business magnate and investor.", "He was the chairman, chief executive officer (CEO), and co-founder of Apple
Inc.; chairman and majority shareholder of Pixar; a member of The Walt Disney Company's board of directors
following its acquisition of Pixar; and the founder, chairman, and CEO of NeXT.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": "Steven Paul Jobs",
                                "2": "Pixar",
                                "3": "chairman",
                                "4": ""
                        }
                },
```

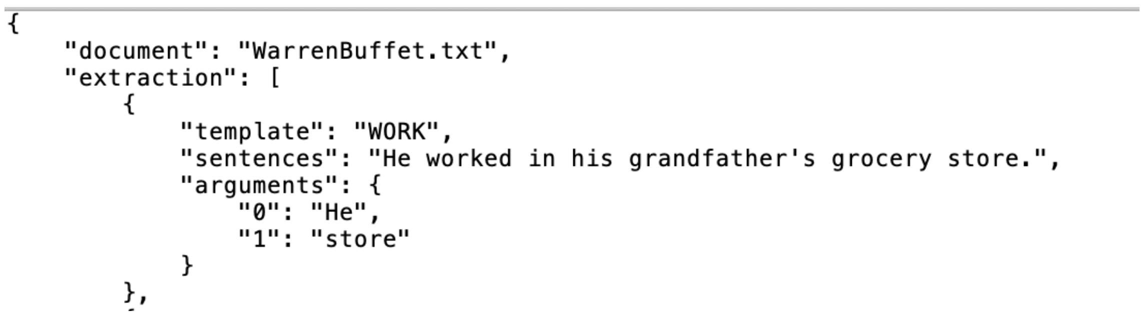
PART



```
        {
            "template": "PART",
            "sentences": "At age six, Winfrey moved to an inner-city neighborhood in
Milwaukee, Wisconsin, with her mother, who was less supportive and encouraging than her
grandmother had been, largely as a result of the long hours she worked as a maid.",
            "arguments": {
                "0": "Milwaukee",
                "1": "Wisconsin"
            }
        },
        {
            "template": "PART",
            "sentences": "Winfrey currently lives on \"The Promised Land\", her 42-acre
(17-ha) estate with ocean and mountain views in Montecito, California.",
            "arguments": {
                "0": "Montecito",
                "1": "California"
            }
        },
```

Some of the problems we noticed during the course of this project is that the relationship extraction is hugely dependent on the quality of coreference resolution done. If the coreference resolution is not done well, then we get entries as shown below:

Problem 1



```
{
    "document": "WarrenBuffet.txt",
    "extraction": [
        {
            "template": "WORK",
            "sentences": "He worked in his grandfather's grocery store.",
            "arguments": {
                "0": "He",
                "1": "store"
            }
        },
```

As you can see here, instead of identification of the subject ”John”, our system incorrectly identifies ”he”as the subject.

We resolved this using Stanford’s Corefernce Resolution system which helps reduce such cases in the output to a certain extent.

Another significant problem that arises is that all named entities in a sentence are sometimes not tagged properly by the StanfordNERtagger. This creates problem in technniques where we use named entities for further processing. We solved this problem by using extracting key phrases and checking for proper Noun (NNP, NNS) in phrases to make sure that the named entities didnt leave out an important entity.

I think overall the project was quite interesting and I learnt a lot about handling textual data and how to derive meaning and structure from it.

Referências

[1] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky - *The Stanford CoreNLP Natural Language Processing Toolkit* In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60, 2014.

[2] Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. - *Leveraging Linguistic Structure For Open Domain Information Extraction.*, In Proceedings of the Association of Computational Linguistics (ACL), 2015.