# CMPUT 690: Assignment 1

**Peng Xu,** *pxu4@ualberta.ca*

## 1. Preprocess

- Split the original document into 826 separate documents, which can be found in the folder named with "data".
- Replace '\\n' with '\n' in the original text.
- Source code can be found in 'src/split.py'.

## 2. Baseline: Stanford NER Tagger

Via NLTK package in python, we can easily access Stanford NER Tagger. After tokenizing the documents, we can get the name entity recognition result with the tagger. With a little manipulation, we can get our baseline result with three classes.

From the baseline method, we get 4238 instances and 3 classes (PERSON, ORGANIZATION, LOCATION). Although there are some mistakes, the result is fairly good. And we will identify more classes based on the baseline result.

In the implementation, Unicode should be considered carefully to avoid annoying bugs. The source code can be found in 'src/ner.py' and the result can be found in 'output/baseline.txt'.

## 3. Improvement 1: "is/are" Relation

One intuitive idea to identify more classes is to detect the relations in the context. However, the sematic relations are not easy to capture due to the intricacy of syntax. Inspired by the idea in "Extracting Semantic Concept Relations from Wikipedia", we can use manually defined sematic relation patterns to detect the sematic relations.

In the article mentioned above, finite state machine (FSM) is used to identify the sematic relation patterns. However, it's a little bit complicated to code. After a little testing, we find that "is/are" relation can be found in the first sentence in all documents we have and there are mainly three types: soccer player, soccer team and stadium. Based on these observations, we can have a simpler method to capture the sematic relation in the first sentence of the documents.

First, we define three category of words for the corresponding classes:

- **soccer_player**: footballer, player, defender, forward, midfielder, striker, goalkeeper
- **soccer_team**: club, team
- **stadium**: stadium

Then, we need to locate the verb indicating the "is/are" relation, like 'is', 'are', 'was' and so on. According to the location of the verb, we can identify instances before the verb and find target words after the verb. More specifically, we identify instances based on the baseline result; we use Part-of-Speech (POS) Tagger in NLTK to tag the first sentence and find the nouns after the verb. If we find target nouns in the pre-defined category and it matches the baseline class, like **soccer_player** matches "PERSON", we can attach class "soccer_player" to this instance. Figure 1 illustrates this process.

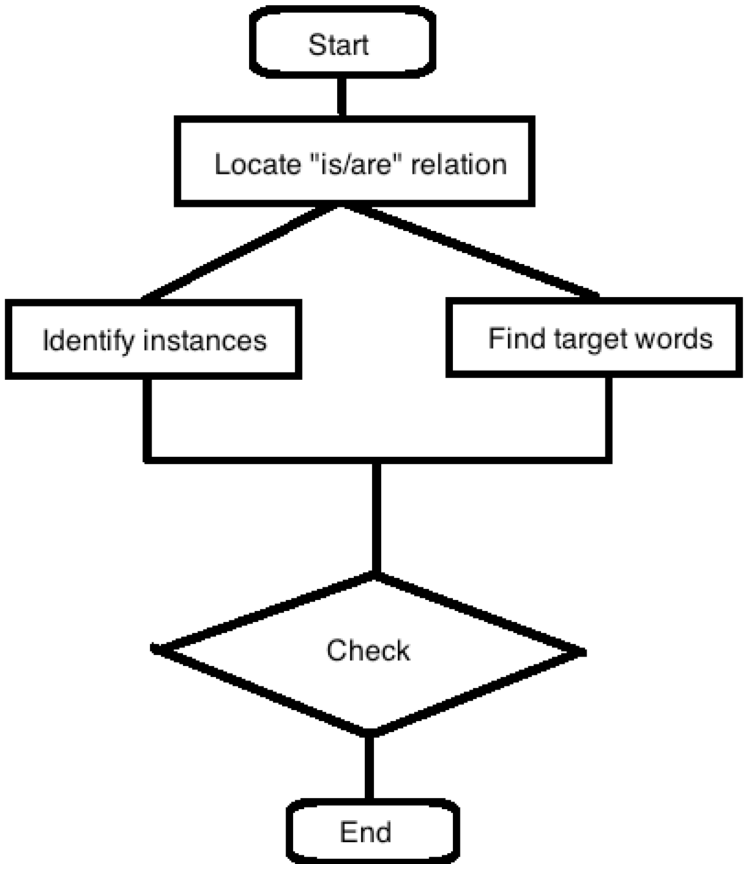

Figure 1

## 4. Improvement 2: Wordnet

There is no doubt that Wordnet is a powerful tool. But how can we use it in our task? It's not hard to find that there are many location instances extracted by NER can be found in Wordnet, like "Paris" and "England". However, there are no explicit indications of its class, like city, country and state, in the context. With the help of Wordnet, we can give class to location instances obtained by the baseline method.

To be more specific, we access Wordnet through NLTK and make use of the definition text of the corresponding synsets. Then, identify words like 'city', 'captial', 'country', 'republic' and so on. According to some rules, we can give 'City', 'Country', 'State' class to the corresponding instance. For example, the definition text of Synset('paris.n.01') is "the capital and largest city of France; and international center of culture and commerce". Due to the word "capital", we can give class 'city' to the instance 'Paris'.

## 5. Bonus: Coreference Resolution

We find that there are many coreference instances in the first sentence, through the words like 'simply', 'refered as'. Based on a very simple method, we can get fairly good results without delving into the syntax. The method is that if two entities extracted in the first sentence have the same baseline class and their Jaccard Similarity is not zero, then they refer to the same instance.

The result turns out to be fairly good with high accuracy. The source code can be found in 'src/CR.py' and the coreference resolution result can be found in 'output/CR.txt'.

## 6. Issues Encountered

- I also want to use some hot techniques like Word2Vec to solve the problem of coreference resolution. However, probably due to the size of corpus, the result is not good. The source code can be found in 'src/Word2Vec.py' and the trained Word2Vec model is saved in 'model/word2vec.1'.
- All our work is based on the result of the baseline method. It causes that the mistakes in the baseline method cannot be corrected. And during the subsequent process, these mistakes will accumulate, which is not desired.

## 7. Appendix

**Class Detected**

- Class extracted by Stanford NER
  - PERSON
  - ORGANIZATION
  - LOCTION
- Class identified by "is/are" relation
  - Soccer_player
  - Soccer_team
  - Stadium
- Class identified by wordnet
  - City
  - Country
  - State

**Statistics on Final Results**

- 9 classes, 5246 instances
- 55 coreference resolution groups