Eric Cheung
0362519

Solution Summary

The goal of this assignment is to identify relations between pairs of entities in natural language text by applying basic Natural Language Processing techniques. First, I visually inspected the source document to look for common reoccurring patterns. And since the content of the file were extracted from the latest dump of Freebase describing organizations, people, and locations, related to professional football, we can devise methods to take advantage of this assumption. From the last assignment, we used the concept of *anchor* term [1] that commonly exists in the first sentence of a Wikipedia type of documents that identify the key *subject* of the piece. This solution reuse this idea and code base from assignment 1 with some code enhancement. It continues to use the Apache OpenNLP v1.6 library for the sentence processing and POS (Part-Of-Speech) tagging, then apply FSM (Finite-State-Machine) for pattern matching. The library also offers several ready-to-use classifiers for the classes of Person, Location, and Organization, which we are using in this solution. In addition to identifying hyponym and synonym in the documents, more FSMs were created for finding relationships between entities. An implementation of a Triple-Store was created to temporary store all the relationships in the form of triples. It also stores the pre-loaded predicates identified during the visual inspection step. Simple co-reference technique were used based on the anchor and checking for entity class before applying certain relationships. For example, the pre-condition for accepting the "<subject> plays-for <object>" pattern is that the <subject> must first be identified as a *person* class. Formatted reports are generated using the content in the triple store. In total, the solution extracted 1387 relationships from provided input text file, it also a second output file in respond to the bonus question. Figure 1 depicts the high level framework of the solution.

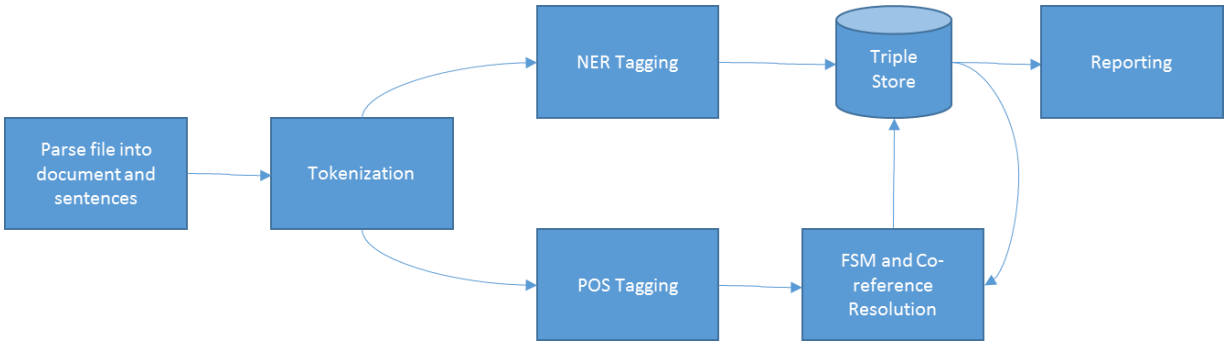


Figure 1 Document Processing Framework

The solution uses an OOP model with multiple classes serving different functions.

Classes & Files	Description
config.properties	This is a configuration file used to modify solution's behaviour without recompiling the code base. Simply change the content using any text editor and re-run the solution.
predicates.tsv	This is an input file containing pre-identified predicates that FSMs are associated with. This file also create a mapping between predicates. For example, the patterns: *played-for*, *debuted-for*, *started-at* are all associated with the relationship *played-in-team.* This linkage is useful when producing the summary report on the frequency and source of each patterns based on groupings of relationships.

Engine	This class implements the pipeline of the traffic where handles the traffic and steps involved. It deals directly with the OpenNLP library objects to use its services. It creates instances of the OpenNLP classes and make use of the *tokenizer* (turning a sentence into an array of words), POS tagger (applying part-of-speech tags), and triggering the NER classifier models to find Persons, Organizations, and Location entities. It also extracts entities using the FSM class and perform simple co-reference resolution, then store the relationships into the TripleStore instance.
Start	The entry class for the solution. It contains the main method that users could call to start the extraction process. It creates instances of and makes use of the other classes. It is responsible for creating input stream of the input data and models and directs the flow of the program.
FSM	This static class implements the various Finite-State-Machines that can matches patterns based on simple Hearst and text patterns. The patterns that are currently supported by the FSM class are: *Born-In*, *Played-Against*, *Plays-For*, *Started-At*, *City-Of*, *Debut-With*, *Lived-In*, *Founded-By*, *As-A-Role*, *Known-As*, *Is-A*.
Triple	A small class for carrying a *triple* (Subject-Predicate-Object). It is often used as part of a vector list returned by the FSM.
TripleStore	A data store of *triples*. It provides triple storage and retrieval services. It can also generate formatted summary report using its content.

Issues

Although many relationships were successfully identified by the solutions, there are some areas that could be improved.

Inconsistent POS Tagging – The solution is dependent on the Apache OpenNLP library for providing POS Tagging functionality. For the most part it does a great job in providing accurate tags that is essential for the custom FSMs to work. However I do notice odd instances of inconsistency when identifying nouns. For instance, the tagger identifies the term "Brazilian footballer" as Adjective [JJ] and Noun [NN], while the term "Uruguayan footballer" is tagged as Proper Noun [NNP] and Noun [NN]. Inconsistency such as this pose a problem for the FSM that depends on the [NN] tag for identifying subjects and objects. I also experimented with the Stanford POS tagging and NER classifier, but did not fully implement it to the solution due to time limit as well as run-time issues. I find the Apache OpenNLP to be much simpler and lightweight in comparison.

Brittle FSM pattern matching – The solution uses FSM for pattern matching, this approach can be considered brittle because the matching conditions can be very straight where either a matching is found or not according to the design. Also, a lot of manual effort is required to maintain the design.

Entity Disambiguation – The solution currently utilize the FSM for detecting entity synonym within a document. This works reasonability well within the same document. However, if the same literal are found in another document, the solution logic currently simply reuse identify it as the same entity without performing additional disambiguation resolution.

Incomplete Coreference Resolution – The *Engine* class is responsible for resolving coreference and it is under developed. The main assumption made is that the anchor term is used as the main subject, and the relationships are referring to it as the subject. However, the FSM had been enhanced to return last

mentioned object as well. This information could be used for co-referencing but it is not fully implemented at this point. It currently applies a sanity check before associating a subject to a predicate, for example, only a person entity could be used as the subject of the "born-in" triple.

Areas of Improvement

There are many areas in the solution can be improved if given the opportunity. A better NER classifier that had been trained with similar data set may help in identifying named entities right off the bat. In the area of improving agility of the solution, *Parsing* could be used to break the sentences into different noun phrases using treebank format, then different mentions could linked to relationships that within the phrase. This could help with the co-reference resolution where the FSM method has difficult with. Using a lexical database, such as the WordNet, in combination with Hearst Pattern matching, the solution may be able to provide a more dynamic method of identifying relationships than the current FSM method. Linking to external knowledge base could help with entity disambiguation, as well as class verification to increase the accuracy of the relationships.

Reference

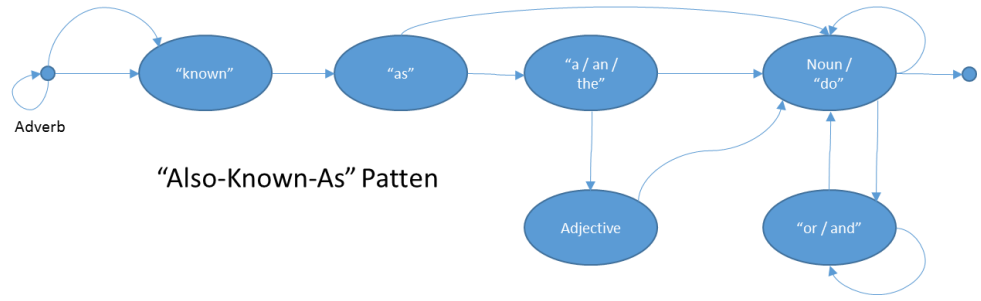
1. Extracting Semantic Concept Relations from Wikipedia. Arnold, Patrick and Rahm, Erhard. In Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14), pp. 26:1-26:11. ACM, New York, NY, USA, 2014.

2. "General." Apache OpenNLP. Web. 06 Feb. 2016. <https://opennlp.apache.org/>.

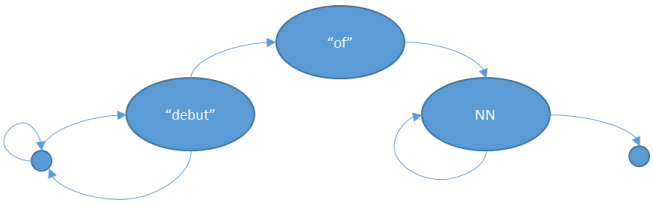
3. Princeton University "About WordNet." WordNet. Princeton University. 2010. <http://wordnet.princeton.edu>.

Appendix

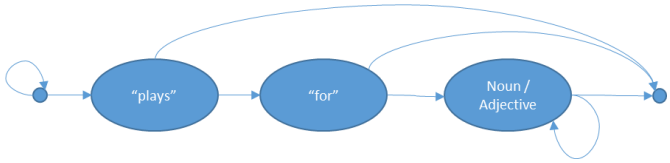
Below are some examples of the Finite-State-Machines used in the solution:



"Also-Known-As" Patten



"Debut for"



"Plays For" pattern