Solution Summary

In order to extract meaningful entities and classes from the source file, I first visually inspected the document to determine the content and the context it is in. The content of the file were extracted from the latest dump of Freebase describing organizations, people, and locations, related to professional football. Each paragraph within the file represent separate "document", where lines within a document are delimited by the carriage-return character. I implemented the solution with the Java Standard library (1.7) using ideas from [1]. Utilizing the Apache OpenNLP v1.6 library for sentence processing and POS (Part-Of-Speech) tagging, and FSM (Finite-State-Machine) for matching patterns, then extracting meaningful entities and classes to create *triples*. The OpenNLP library also offers several ready-to-use NER (Named Entity Recognizer) models for classifying classes like Person, Location, Organization, etc. But in order to find specific classes that the NER could not identify, I implemented several FSMs by applying Hearst patterns to find semantic relationships unique to the context and structure of the data file. The solution uses an OOP (Object Oriented Programming) model with multiple classes providing different functions. Table 1 summarize the classes' functions.

Java 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.
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 produce the results as output files. This is the nerve centre of the solution that connects everything together.
Engine	This class deals directly with the OpenNLP library objects on behalf of the main solution. 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. This is the brain of the solution.
Document	A class used to store documents. It contains a series of Sentences objects.
Sentence	A class used to store sentences within a document. It also contains the FSMs for matching Hearst pattern that return *Triple* objects.
Triple	A small class for carrying a *triple* (Subject-Predicate-Object).
TripleStore	A data store of *triples*. It provides triple storage and retrieval services. The heart of the solution.

Table 1 Class and File Summary

From [1], we learned that the first sentence of the description in a wiki page often contains very useful information about the subject being discussed. Since the provided data was captured from Freebase, whose data was harvested mainly from Wikipedia, my solution uses this knowledge to extract the *Anchor Term* from the first sentence. This anchor term is used to build Triples for the remaining of the documents as it is often the core subject being referred to. Below is a list of the FSMs implemented in the solution (See the Appendix for graphical representations).

"Is-A" Pattern – This FSM is applied to the first sentence of the document. The main goal is to find the Anchor term and the class it belongs to. For example, "Franco Mussis is an Argentinian football player" would match the pattern where "Franco Mussis" is the subject while the "football player" is the object. The

extracted triple is then stored into the triple store. The solution also infers the relationship that a "football player" is a subclass of the "person" class.

"Also-Known-As" Pattern – This FSM is applied to the portion of text between commas within the first sentence. From visual inspection, it appears to be a common pattern that the "also-known-as" pattern appears quite often. The solution uses this knowledge to extracts synonym relationships.

"Plays-For" Pattern – This FSM is applied to all sentences in a document to find the relationship between a player (anchor term) and a football organization. When found, the organization is stored as a football club and organization.

"As-A" Pattern – This FSM is applied to all sentences in a document to find out what field positions a player (anchor term) usually occupy. When found, the object is tagged as a "position" class and the player is tagged as a subclass.

"City-Of" Pattern – This FSM is applied to all sentences in a document to find potential city names. When found, the object is tagged under the "city" class.

In addition to the FSMs, the solution also uses three of the NER models provided to find Person, Organization, and Location class instances. All Triples generated from the FSMs and NER models are stored in the TripleStore for additional processing and retrieval as needed. Outputs of the primary objective, hyponym relationships, are generate relatively easily by querying the TripleStore for all triples that has the "is-a" as predicate. As for the bonus objective of finding synonyms, rudimentary nested loops are used to find all synonym for each particular subject and the output are stored in another text file.

Issues

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]. Cases like this pose a problem for the FSM that depends on the [NN] tag for identifying subjects and objects.

Erroneous Classification – Although the solution is able to find many instances of the hyponym relationship (3380), there are some erroneous classification. In the sentence "Due only playing the sport as a hobby", the player Adil Rami was classified as a "hobby" by the "As-A" pattern matching FSM. Although rare, but this poses a question on balancing between accuracy and recall for the solution.

Entity Disambiguation – The biggest design issue encountered while developing the solution is how to handle the literal of the entities. Currently the solution makes the assumption that the entity literal are unique, thus if other entities with the same literal are found, the same entity ID will be used. This flaw can be seen in the bonus output where multiple players have similar nick name "Javi" and thus the algorithm erroneous grouped them as synonymous of each other which is incorrect. This is a bug that need to be corrected in future releases.

Contextual Dependency – As one may notice, some of the FSMs are designed to fit the context of the documents provided, this is a problem if the documents have changed topic. I believe this is an acceptable trade-off between speciality and generality of the tool as long as we are aware of the context the solution is applied to.

Out-Of-Memory Error – This problem arise when running on computer with smaller memory allocation (<1GB). To combat this issue, I added the starting memory argument of 1024MB to the ant script when executing the program.

Areas of Improvement

There are many things can be done to improve the performance and design of the solution. Here are main areas of improvement should be tackled next:

Entity Disambiguation – In order to improve the current situation of identifying entities with same literal, the assumption of treating literal as unique should be removed. This can be done by creating a special predicate such as "has literal" with the entity ID as the subject and a literal string as the object. However, the solutions than need to consider when to and not to merge similar looking entities.

Generalize FSM – The current sets of FSMs are heavily contextually dependent and would not work well when applied to a different topic. Better use of Hearst Pattern will be needed to improve generalization of the matching patterns.

Better POS – Use of Noun Phrases and Chunking into a treebank may be considered to help with entity identification.

Classes

The OpenNLP library enabled us to classify three core types of classes, Person, Organization, and Location. In addition, our FSMs are able to identify several different classes related to the documents topics:

Class	Description
Person	The super class for identifiable individuals. It can be generated by the both the NER classifier or the "Is-A" FSM.
Organization	The super class for types of football team or associations. It can be generated by both the NER classifier or the "Plays-For" FSM.
Location	The super class for all types of places such as cities or countries. It can be generated by both the NER classifier or the "City-Of" FSM.
Football_team	Football_Team is a subclass of organization and is generated by the "Plays-For" FSM.
City	City is a subclass of Location and is generated by the "City-Of" FSM.
Position	Position is class encapsulating football field positions, such as Center, Midfield, etc. They are identified using the "Play-As" FSM.
Nationality	The special class used to capture nationality of the players. It is captured from the "Is-A" FSM as the Adjective that frequently appears before the term "footballer". For example, "Roberto Vitiello is an *Italian* footballer".
Others	The solution also attempts to create new classes based on the positions, such as "goalkeeper", "winger" from the "Is-A" FSM.

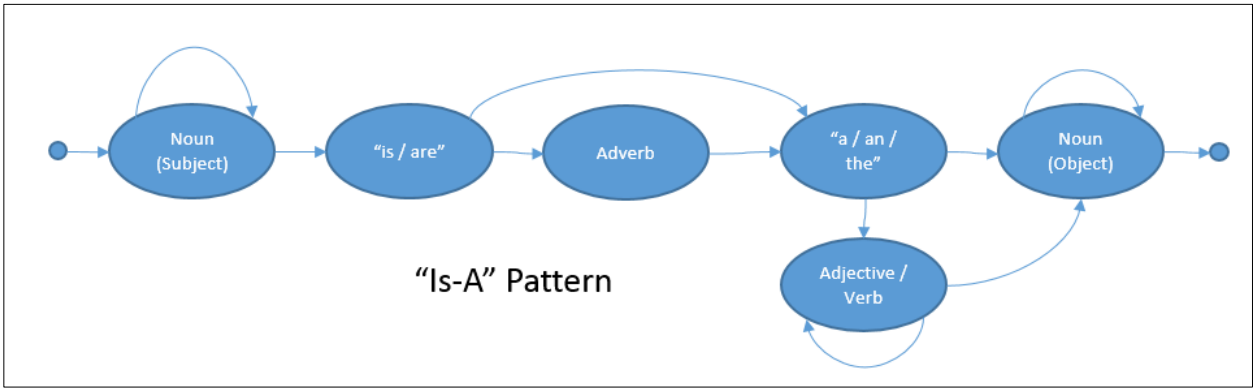
Table 2 List of Classes the solution can identify.

Reference

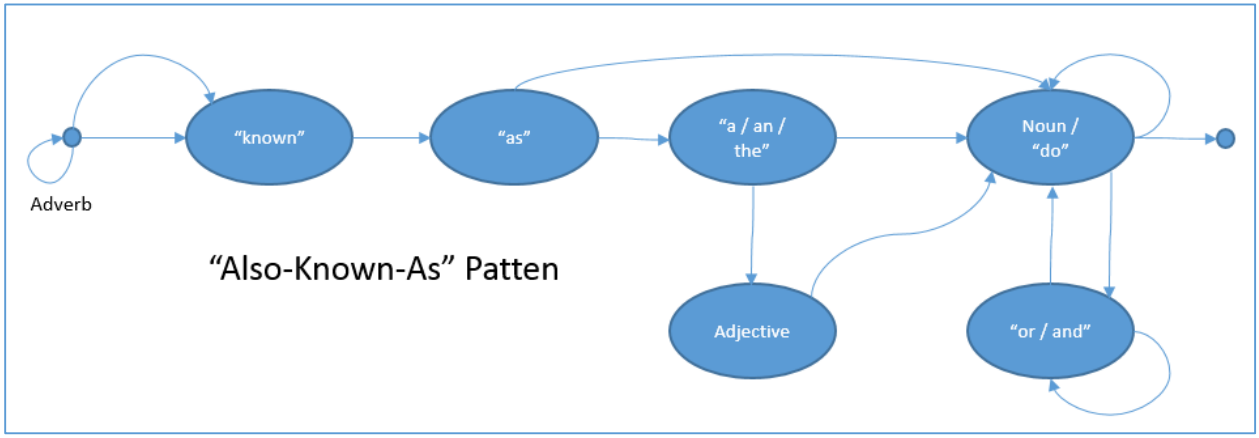
1. Extracting Semantic Concept Relations from Wikipedia. Arnold, Patrick and Rahm, Erhard. In Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14), pp. 26:1-26:11. ACM, New York, NY, USA, 2014.
2. "General." Apache OpenNLP. Web. 06 Feb. 2016. <https://opennlp.apache.org/>.

Eric Cheung
0362519

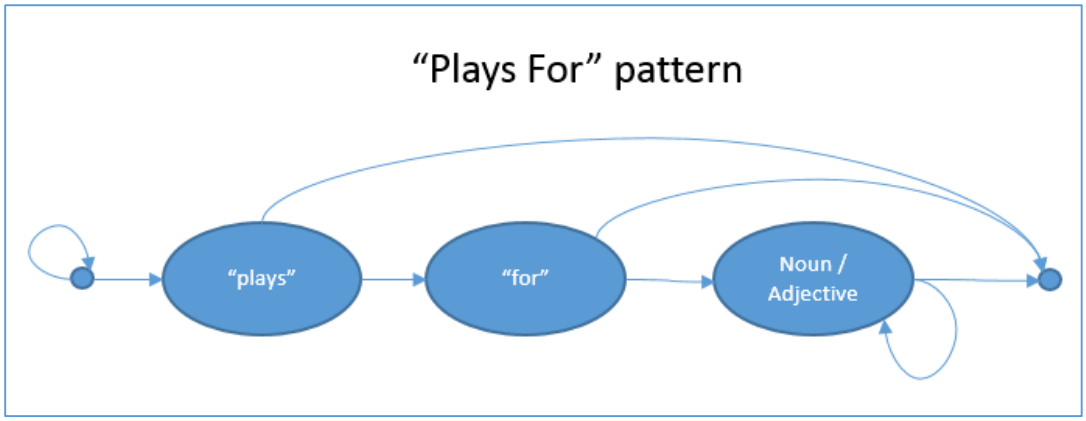
Appendix



The "Is-A" pattern produce the hyponym relationship.

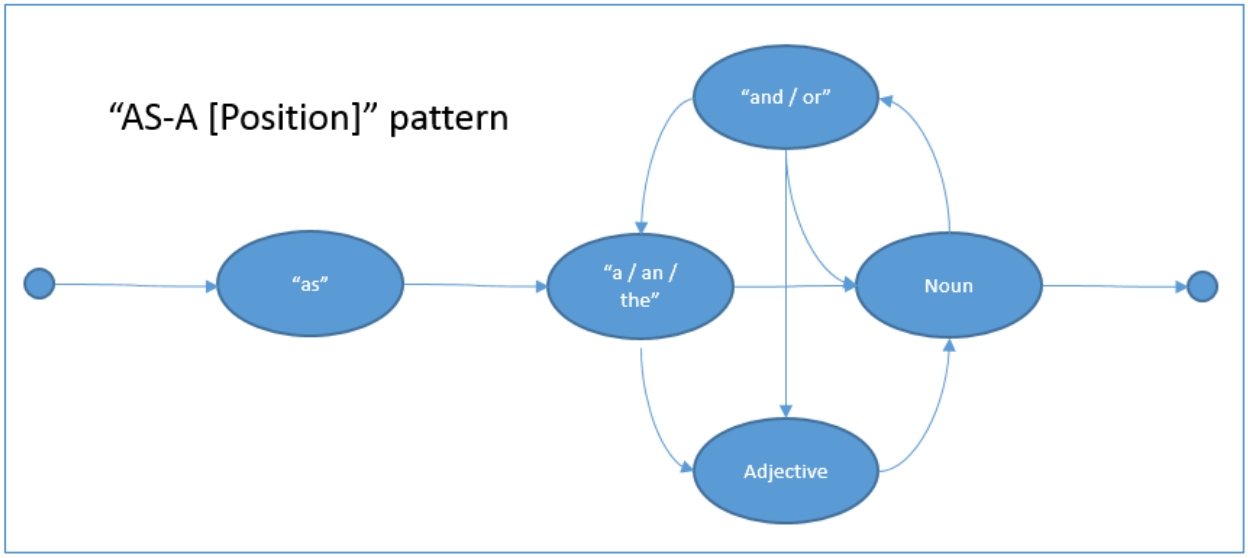


The "Also-Known-As" pattern captures synonym relationship between subjects.

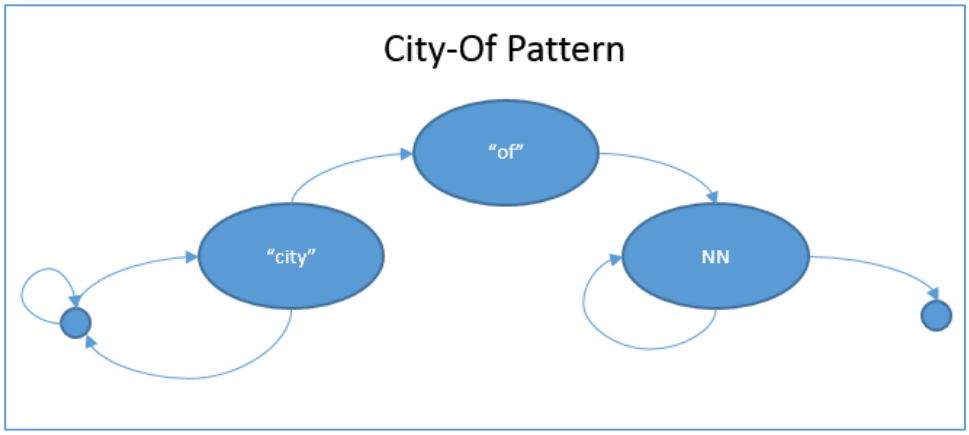


The "Plays-For" pattern captures an organization entity.

Eric Cheung
0362519



The "As-A" pattern captures the player's typical position in a football game.



The "City-Of" pattern captures city name as subject.