Entity Recognition & Relationship Extraction With CoreNLP

Yoav Ilan, Danyaal Ali, Shray Mittal, Manisha Gupta

Background

Entity recognition and relationship extraction are fundamental parts of natural language processing, as it allows the possibility of efficient and effective information extraction. As the corpus of written natural language steadily expands, effective automated tools that can reliably find named entities and relationships between them grow in importance. Since these processes are such foundational pieces of the process, there are several open-source libraries available to complete these tasks. We found SpaCy and CoreNLP to be the most useful in identifying named entities and extracting relationships between those entities.

Problem Statement

We were given the task of analyzing a database of our choosing to find specific relationships between named entities in an effort to create a graph-like model of connected entities. We chose to analyze a corpus of Wikipedia articles about the players of the Premier League to find named entities and the relationships between them. Specifically, we trained CoreNLP machine reading for <player> plays for <team> and <player> is <nationality> relationships.

Strategy and Implementation

Finding Named Entities

SpaCy turned out to be extremely useful in identifying named entities. After feeding it our corpus of soccer players and teams, it picked up a solid portion of the named entities we wanted. Still, the labels that were produced by SpaCy were a very general, in that a team would be labeled as ORG or GPE, which was the same labels a city would receive.

Looking back on what we did in this project, taking time to work on named entity extraction seems extraneous. Once we got to working with CoreNLP, we realized that the system automatically did that when finding relations, and we could filter our relationship results to only include ones with relevant named entity tags associated with the entities.

Resolving Co-references

We explored the route of resolving co-references throughout or corpus in an attempt to improve the relations that CoreNLP could pick up on. We were able to configure SpaCy to be able to resolve the pronouns referring to specific clusters in the document, but the results we came out with were not perfect. For example, we were able to find and replace instances where the player “Petr Čech” was referred to as “he”. However, in sentences where the player was not the direct subject, like in “During his eleven-year stay at the club, Čech registered” the system would determine the subject of the sentence to be “his” instead of “Čech.” SpaCy would then replace the succeeding instances of “he” with “his,” which proved to be more harmful to our dataset. Because this was a fundamental problem in SpaCy’s co-referencing system, there was not much modification we could do on this end.

Extracting Relationships Between Named Entities

This was easily the most difficult and time-consuming portion of the project. There are so many relationship extraction systems out there with so little information on them that it made it difficult to pick the one that would work best for our purpose. Originally, we tried to use OpenIE, but that didn't work out the way that we hoped it would, as explained in the problems we faced section, so we decided to go in a different direction. After one of the class presentations, we decided that using CoreNLP was the way to go. We decided to train the system to recognize the custom relationships that we wanted to extract, which was a long and difficult process, but after doing so we were finally seeing results that resembled our aim. After doing more training and testing our system, we got to a point where we decided that our results were acceptable for the scope of the project.

Results

After training our model and running it on the corpus, we managed to create a well-connected Neo4J database including 300 players, 23 nationalities, and 53 teams. These results were somewhat surprising as we were expecting more players and fewer teams. The increase in teams occurred because other teams were listed in many of the articles and our system struggled to understand that they were not a team that a given player played for, but rather just played against of was mentioned with. The decrease in the amount of players was likely due to our training file being too shallow. It was difficult to find a balance between a large training file and just putting every player in the training file. Also, the trained model was definitely not perfect, as player-team relationships that we specifically trained were not caught when running the relationship extractor on the corpus. We couldn't really do much about this issue, and we had to live with this limitation, given our time constraints.

Problems We Faced

When starting out, we decided to try to use OpenIE in order to do our relationship extraction. Although it was an extremely fast and efficient extraction system, the relationships it produced were not what we were looking for. Often, it made an entire sentence into an entity or a relation, making results quite inaccurate. Also, although it gave lots of results, not one of them was one that we were targeting. Since this platform was used to perform open information extraction, the relations we got were far too general to be able to be useful in this context. Thus, we moved onto traditional custom information extraction.

CoreNLP was a much better choice for us. It allowed us to create custom relationships that were far better for the purpose of this project. However, it came with lots of expenses. Trying to train the system to pick up the relations we wanted was extremely tedious and difficult. The lack of documentation made the whole process exceptionally challenging. When we finally figured out how to train data, we realized how long it would take to accomplish. The training file format was complicated and took a long time to create, and making a training file large enough to make the system as accurate as we wanted was simply impossible with the time constraints we faced (the

original Roth training file included thousands of sentences to train on). Furthermore, retraining data was a complete pain and honestly didn't even work, so when we wanted to update the model, we had to download a fresh copy of CoreNLP and redo everything from scratch. Also, the model struggled with false positives. That is, it often found a team that was mentioned in the text that had no relation to the player himself and wrongly created a plays_for relation between the two entities. We tried to address this by creating a custom NER model, but we found it only worsened our results as it wiped the pre-created model instead of appending to it.

Another unintentional problem we came across was that the corpus we chose was filled with a lot of European cities and player names. As a result, we believe that CoreNLP had a tougher time picking up on some of the relationships since names were not in English. Although this was not a major issue, it could have led to potential mixed relations where the system thought to label a player as a city because of its unnatural spelling in accordance with the English language.

Our final issue also had to do with time constraints. When setting out to do this project, we aimed to have player-team relationships, player-nationality relationships, and team-competition relationships. After realizing how hard it is to train the system, though, we decided that accomplishing all of that was quite ambitious, so we ended up focusing only on relationships involving players.

Overall, CoreNLP was by far the most robust system we found to do what we wanted, but it was also extremely painful to use. The amount of manual training required to create an accurate model is understandable yet simultaneously extreme, especially when working on a tight schedule.

Lessons Learned

After doing this project, we learned a lot about named entity and relationship extraction. Seeing the size of the codebase of these systems like CoreNLP was mind-blowing. It's crazy to realize that people built this extremely complex system that has the ability to work similarly to the human mind. We also learned, through trying to do it ourselves, how complicated it is to train an AI system like this to recognize and pinpoint specific information within a large database of information. Several tools are able to perform these tasks well in the general case, however, extending those tools to specifically pick up on certain relationships was very difficult. We learned how much training and modification is needed on pre-existing tools to be able to perform the actions we like to see.

Other than the NLP portion of the project, we also learned a lot about extracting and cleaning up documents programmatically, as we had to scrape out the players that play for each team and then fetch their documents as well. Finally, we were introduced to graph databases, and by playing around with Neo4J, we were convinced of their benefits to accurately and quickly find relations between nodes, a task that is much less trivial given a SQL database.

Conclusion

When entering this project, we thought that it would not be too difficult. Looking back on all that went into it, though, we were extremely wrong. Not having any background with any of these systems made it extremely challenging to figure out how to use them efficiently. We spent a lot of time going in a direction that--if we had known more about the software we were using--we would've avoided completely. Then, when we figured out the right path to follow, we were extremely overwhelmed by the amount of configuration and training work that was required. Doing our best to accomplish the task, we managed to extract some relationships we wanted with decent accuracy, although we struggled with eliminating false positives.

We believe the next step for this project would be to train CoreNLP to find more relationships, with greater accuracy. Specifically, we would like to search for the "player plays X position" relationship. More training data could be useful in eliminating many of our false positive relationships found, and we could also explore techniques like bootstrapped pattern learning for improving our accuracy. Another potential path of exploration we could take in the future is altering SpaCy's co-referencing model. If we are able to accurately replace all the co-references referring to named entities, we believe our relationship extraction model will be able to come closer to picking out more comprehensive and meaningful relationships.