Inferring Family Relationships from a Corpus

Axel Larsson
Department of Computer Science
Faculty of Engineering LTH
dat11al1@student.lu.se

Erik Gärtner
Department of Computer Science
Faculty of Engineering LTH
ine11ega@student.lu.se

Abstract

This article discusses an experimental method for extracting family relationships from a corpus. In the end the article concludes that the method, while not very accurate at the moment, shows great promise.

1 Introduction

In this article we present our work on inferring family relationships from a corpus. The goal is to extract family relationships between persons found in the text and present them in a graph. We choose to support the same type of relationships as defined by Wikidata (Wikidata, 2016). The application takes a novel as input and outputs a directed graph containing all the persons in the novel as vertices and the relationships as edges.

The work so far is an experimental proof-of-concept application. The system has been designed to easily support multiple languages with English and Swedish being the primary focus. However, at this stage, only English input is fully supported. This is due to the fact that the system, in its current incarnation, heavily relies on specific Stanford CoreNLP features that only support English, (Manning et al., 2014), such as OpenIE (Gabor Angeli and Manning, 2015) and co-reference resolution.

2 Related Work

Google Knowledge Graph is somewhat related to our system, (Google, 2012). Google's system has a huge database of entities and facts with links between the different entities. The main difference with our system, besides the scope, is that our system is a program that, before execution, does not now anything about the contents of the corpus or the persons therein. The Knowledge Graph's input is in the form of semantic queries and the result is an entity matching this query. That result can then have a link to another entity, potentially in the form of a relationships similar to what our system tries to extract such as parents and siblings. Our system on the other hand takes a whole corpus as input and outputs a graph of relationships between persons in the corpus. Moreover, the entities in Google's Knowledge Graph are often famous people while our system tries to find all relationships in the input without any prior knowledge of what entities even exist in the corpus.

Wikidata is another knowledge base that is also used by Knowledge Graph, (Wikidata, 2016). There are named entities in the form of persons in Wikidata that have the exact same relationships that we are trying to find with our program. But again, the difference is that these are all known persons. Our system tries to recognise all persons in the input but does not try and link them to previously known persons, even if some of the fictional characters that the system finds likely could be found in Wikidata.

Stanford CoreNLP have an annotator named *RelationExtractorAnnotator*, however it works on a very limited set of relationships such as *Live_In* and *Located_In* and does not support the type of relationships that we want to find with our system.

3 Detecting Named Entities

The first step of extracting family relationships in a corpus is to identify the characters of the novel. This is done using the Stanford NER (Named Entity Recognition) which is a CRFClassifier that can identify persons, places and organisations. Once all named entities have been extracted we filter out everything except the persons.

Below in Figure 1 we see that the CRFClassifier detects Eric and John as persons, Spain as a country and the U.N. as an organisation. It is these annotation, filtered by type, that are used to extract characters from the corpus.

4 Detecting Family Relationships

We use Stanford CoreNLP's OpenIE annotator. OpenIE or Open Information Extraction is described as "the extraction of structured relation triples from plain text such that the schema does not need to be specified in advance" (Gabor Angeli and Manning, 2015). The structured output consists of so-called open domain relation triples representing the subject, the relation and the object. An example of a triple would be (Miss Turner, is daughter, Mr. Turner), where "Miss Turner" is the subject, "is daughter" the relation and "Mr. Turner" the object.

The software works by splitting the sentence into a set of entailed clauses that are each maximally shortened to produce even shorter sentence fragments; entailed from the original sentence. The fragments are then segmented into OpenIE triples. An example sentence; "Born in a small town, she took the midnight train going anywhere" could be split into a clause "she Bron in small town" which could eventually be segmented into the OpenIE triple "(she; born; town)", see Figure 3.

This annotator is a perfect fit for our system. However, we need to filter the output so that only the relations that we are interested in appear in the output. For instance, we are not interested in the "was born in" relation that OpenIE might extract, but rather relations such as "father to" or "is daughter".

The family relations are defined in a JSON configuration file. The file contains relationship definitions that are represented by a few properties, namely:

- **title** - the title of the relationship, e.g. brother
- **id** - the Wikidata property identifier
- **description** - a description of the property, e.g. male sibling
- **english** - an array of key words indicating the relationship, e.g. ["bro", "brother"]
- **swedish** - the same as above but for Swedish

The **english** and **swedish** arrays of words are perhaps the most interesting ones. This is where the different synonyms for the relationship is defined. By having both a Swedish and English array of words, the found relationship will match across languages, just as it is for Wikidata. The title is used as the label for the edge that is created when a relationship is found.

Title	Description
father	male parent
mother	female parent
brother	male sibling
sister	female sibling
spouse	the subject has the object as their spouse
partner	someone whom the subject is in a relationship without being married
child	offspring
stepfather	husband of the subject's mother, who is not the subject's biological father
stepmother	wife of the subject's father, who is not the subject's biological mother
relative	family member
godparent	person who is godparent of a given person

Table 1: The relationships.

The configuration file specifies the full list of relationship properties that Wikidata uses, see Table 1. The **id** field is mapped to the id of the corresponding Wikidata id, (Wikidata, 2016).

5 Resolving Co-references

Once the OpenIE relation triples have been extracted the subject and object may be co-references such as pronouns. Using the Stanford Co-reference Resolution system implementing multi-pass sieve co-reference resolution pronouns can be resolved into proper nouns.

In Figure 4 the system detects that all the pronouns, he and his, in the sentence refers to Eric.

However it should be noted that resolving is far from trivial. For example if the proper noun isn't in the same paragraph or the sentence structure is more complex then the co-reference resolution may not succeed.

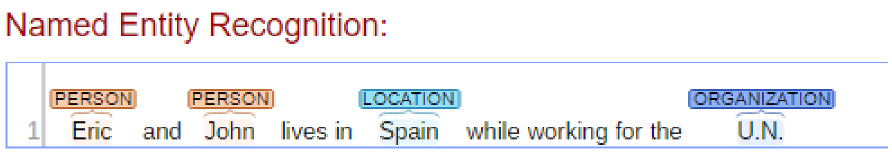
Named Entity Recognition:
PERSON
PERSON
LOCATION
ORGANIZATION
1 Eric and John lives in Spain while working for the U.N.

Figure 1: Pictured is the result of the Stanford NER on a simple sentence.

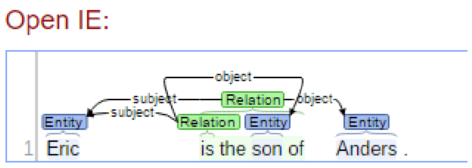

Figure 2: Pictured is the result of the Stanford CoreNLPS's OpenIE on a simple sentence.

6 Inferring Relationships

Once the family relationships mentioned in the text has been identified and connected with named entities a rule engine is used to infer more relationships. For example if the text mentions that "Abel is the son of Adam" then once that relationship has been identified we can infer that Adam is the father of Abel. This way we can extract implicit information from the text that can give a better understanding of the interpersonal relationships in the text.

The idea is to have a set of rules, where some rules depend on the proper execution of other rules. For instance, a gender rule can be used to infer the gender of the entities in the graph. This rule would have to be run before the father-son rule since if we do not know the gender of the subject, we cannot conclude if the subject is a father or mother. With our setup, this type of rules setup is trivial to implement. However, we have not yet implemented a lot of rules. Most of this work is in the early phases.

It might be difficult to specify the order between all rules properly if the number of rules grow. For instance, if the father rule is run it might open up for other rules to infer more relationships such as uncle or grandfather. Thus, we will have the relationship inferrer iterate on the graph, applying all rules at each step, until no more relationships can be found and we have reach some sort of stable graph where all implicit relationships that can be found are found.

7 The Graph

To visualise the found entities and relationships we use a simple directed 2D-graph. Vertices represent persons found in the corpus and the edges represent the relationships between the persons. An example would be a a vertex labelled "Miss Turner" connected with an edge directed at another vertex labelled "Mr Turner". If the edge is labelled "daughter", it would mean that "Miss Turner" is the daughter of "Mr. Turner".

Below in Figure 5 is an example of how that looks.

8 Experimental Setup and Evaluation

While developing and testing the system we used a single novel as a corpus. While it would be preferable to have different training and testing corpora our approach is satisfiable since this is article is about investigating the viability of extracting family relationships. Thus the system is, at this time, a proof-of-concept.

8.1 Corpus

We use the public domain novella *The Boscombe Valley Mystery by Sir Arthur Conan Doyle*, fetched from project Guthenberg, (Doyle, 1999). The novella is 9600 words and contains about ten family relationships. We manually annotated the corpus with all found persons and relationships so we could evaluate the program.

8.2 Results

The results of the relationship detection is shown in Table 2. In the corpus, there are ten relations in total and we are able to positively recall two of those, three are shown to be false positives and the rest are false negatives.

In light of the limited time frame and resources of the project we consider our results to be promising though there are clearly room for improvements.

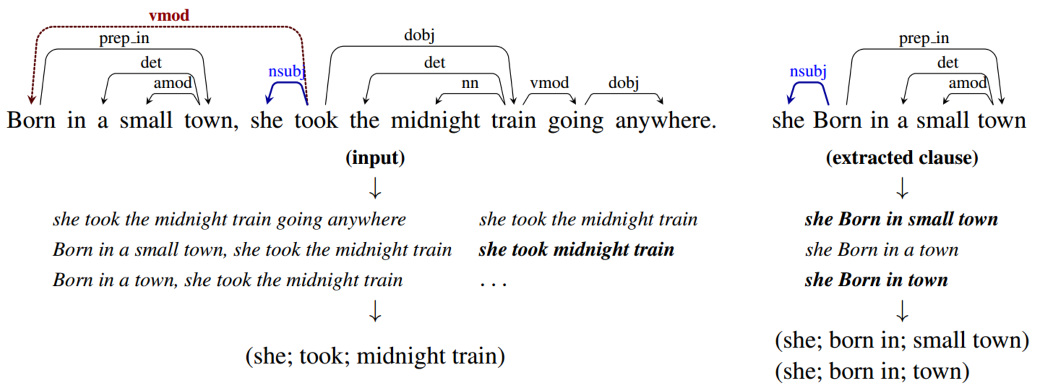
vmod
prep_in
det
amod
nsubj
dobj
det
nn
vmod
dobj
Born in a small town, she took the midnight train going anywhere.
(input)
she took the midnight train going anywhere
Born in a small town, she took the midnight train
Born in a town, she took the midnight train
she took the midnight train
she took midnight train
. . .
(she; took; midnight train)
prep_in
nsubj
det
amod
she Born in a small town
(extracted clause)
she Born in small town
she Born in a town
she Born in town
(she; born in; small town)
(she; born in; town)

Figure 3: A description of Stanford Open Information Extraction module.

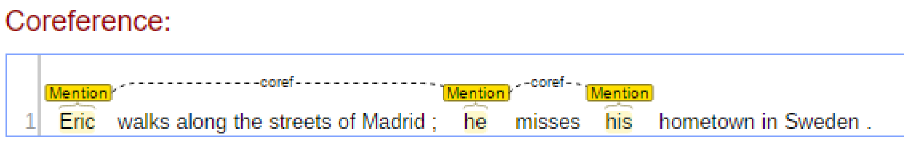
Coreference:
Mention
coref
Mention
coref
Mention
1 Eric walks along the streets of Madrid ; he misses his hometown in Sweden .

Figure 4: Pictured is the result of the Stanford CoreNLPS's Co-reference Resolution system on a simple sentence.

True positives	2
False positives	3
False negatives	7
Recall	0.22
Precision	0.40
F-score	0.29

Table 2: The relationships.

The two main obstacles of the project is accurately determining the subject and object of a relationship and resolving the co-references.

As we see from Table 2 both the precision and recall is around 20%. The low recall stems from the CoreNLP OpenIE failing to extract relevant relationship triples. Improving this part of the program should be the main focus since it acts independently from the named entity recognition, i.e. extracting relationship triples does not use the named entities. Rather the named entities and the entities from OpenIE are matched and merged. The reason behind this method is that named entity recognition won't detect all types of entities that may be interesting. Take the following sentence: "what does the idiot do but get into the clutches of a barmaid in Bristol and marry her at a registry office". CoreNLP named entity recognition will not detect "barmaid in Bristol" as an entity, since it is not a proper noun. Though for our purposes she is still interesting. That is why we merged entities from both OpenIE and NER.

The case of the low precision is also brought on by low accuracy in the OpenIE algorithm since it is common that is incorrectly extracts the subject and the object of the relationship.

There is currently only support for English in the system, while the groundwork to support Swedish is partially laid.

One factor that probably negatively affects the accuracy of the system is the per paragraph splitting; instead of analysing the entire corpus all at once, it is split into paragraphs. This is done to ensure that we are given a result within a reasonable time frame and within reasonable memory and CPU time usage. For instance, one can imagine that certain co-reference chains could be resolved if more paragraphs were analysed at the time since sometimes the naming of a character in the novel is done several paragraphs ahead of a mention indicating a relationship.

The rules engine, while currently very simple, shows promise since it allows for the system to infer more relationships in an iterative process until the set of relationship converges. This functionality could be useful in a system that tries to build a unified knowledge graph by processing many different corpora.

Another big obstacle we encountered was the lack of annotated corpora, that is corpora where

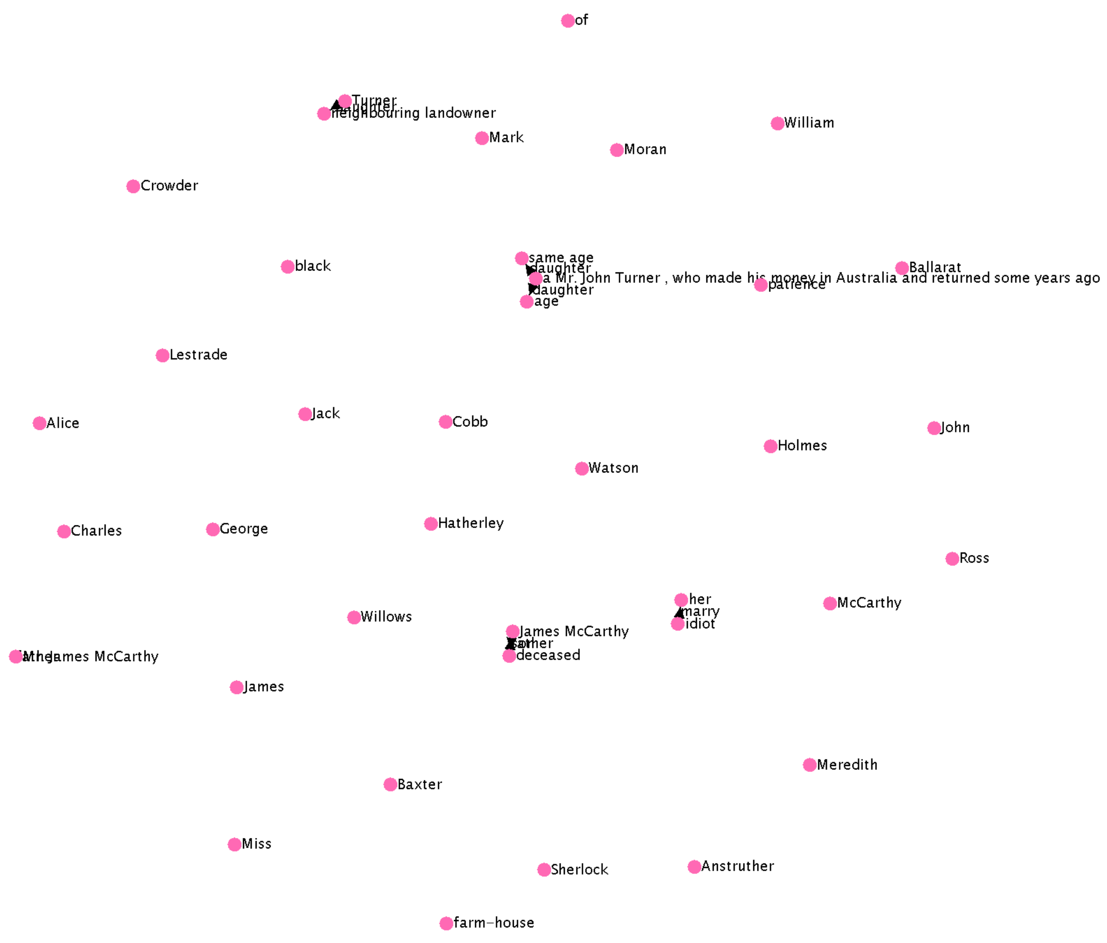

Figure 5: Pictured is the graph generated when analysing "The Boscombe Valley Mystery"

all mentioned relationships had been extracted and listed. Finding novels with a high amount of relationships, reading and then annotating them is time consuming. While it would have been beneficial to have multiple corpora to train and test against it was beyond the time frame of the project.

9 Future Work

While the F-score was low the project shows great promise. Further development should focus on improving recall of the relationships. This can be done by adding new methods such as searching for keywords and then verifying the found sentences by looking at the POS-tags.

Improving the inference engine with more relationships and more sophisticated logic would also be an important next step since it would allow the extraction of more complex relationships.

As previously discussed, OpenIE does not find enough structured relationships to get us a very good result. It that might be possible for us to write our own *family* relationship extractor that only looks for family relationships and not all the structured data that OpenIE tries to detect. A possible approach would be to train some machine learning model on a large corpus of annotated family relationships. Such a trained model might perform better in the specific instance of detecting family relationships.

References

[Doyle1999] Sir Arthur Conan Doyle. 1999. The adventures of sherlock holmes. `https://www.gutenberg.org/ebooks/1661`. Accessed: 2016-05-19.

[Gabor Angeli and Manning2015] Melvin Johnson Premkumar Gabor Angeli and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *In*

Proceedings of the Association of Computational Linguistics (ACL).

[Google2012] Google. 2012. Introducing the knowledge graph: things, not strings. `https://googleblog.blogspot.se/2012/05/introducing-knowledge-graph-things-not.html`. Accessed: 2016-05-11.

[Manning et al.2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

[Wikidata2016] Wikidata. 2016. Relationship properties. `https://www.wikidata.org/wiki/Wikidata:List_of_properties/Person#Relationship`. Accessed: 2016-05-11.