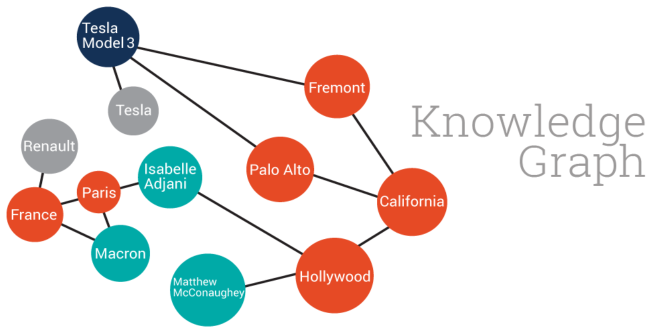


Insights - Knowledge Graph Final Report (Spring 2019)

Contributors: Jenny Chen, Kevin Ngo, Oscar So, Tushar Khan, Ziwei Gu

1 Introduction

1.1 Overview

This project[1] seeks to create structured representation of knowledge based on unstructured text. The end goal of the project is an end-to-end system that generates a knowledge graph from raw text, with the potential to assist users in understanding a source article and facilitate question-answering system (by enhancing searchability over entities in an article). Our resulting knowledge graphs are also natural inputs to Graph Neural Networks, a family of ML models that take graphs as inputs.

Our system yielded substantive results, generating a set of well-formed relation tuples from any input paragraphs. We then convert the tuples to a network graph to aid visualization.

1.2 Background

A knowledge graph (KG) is a graph-structured knowledge base that stores knowledge in the form of the relation between entities (e.g. (LeBron James, plays for, Los Angeles Lakers)). Knowledge Graphs play a critical role in many modern applications, such as question answering and query expansion. Constructing a knowledge graph from unstructured text, however, is a challenging task due to the variety of open relations and the massive ambiguity in text data. Therefore, we focus on exactly these two challenges - extracting open relation triples from raw text and reducing the ambiguities within them.

[1] Github repository: https://github.com/CornellDataScience/Insights-Knowledge-Graphs

2 Methodology

2.1 Overall Architecture

In this section, the overall architecture of our system is described. Our model is designed to take unstructured text as input and produce a knowledge graph as output. As shown in Figure 1, it has the following five components: Triple Extraction, Entity Mapping, Coreference Resolution, Triple Integration, and Predicate Mapping.

As a general overview of the whole pipelined process, we first perform open relation extraction to obtain a set of binary relation tuples, one for each sentence in the input article, and then reduce the ambiguities (pronouns, different references to the same entity, etc.) in the tuples. After extracting all the entities and relations, identical entities are grouped using coreference chains from the Coreference Resolution component. Since entities in a given group might have various representations, a representative for the group of coreferring entities is then selected by a voting algorithm in the Triple Integration component. Empirically, the best voting scheme seems to just choose the majority excluding pronouns as the group representative. After that, all entities belonging to the group found in the relation triples are replaced by this representative, and we map it to either a new ID or an ID in the existing KG. Once we are done with entities, all the relations are vectorized using knowledge graph embedding and the pairwise cosine similarity among all the relation vectors calculated. We then collapse all the similar relations based on a predefined similarity threshold and output the resulting set of triples. Finally, the triples are converted to a specific JSON format to generate a knowledge graph.

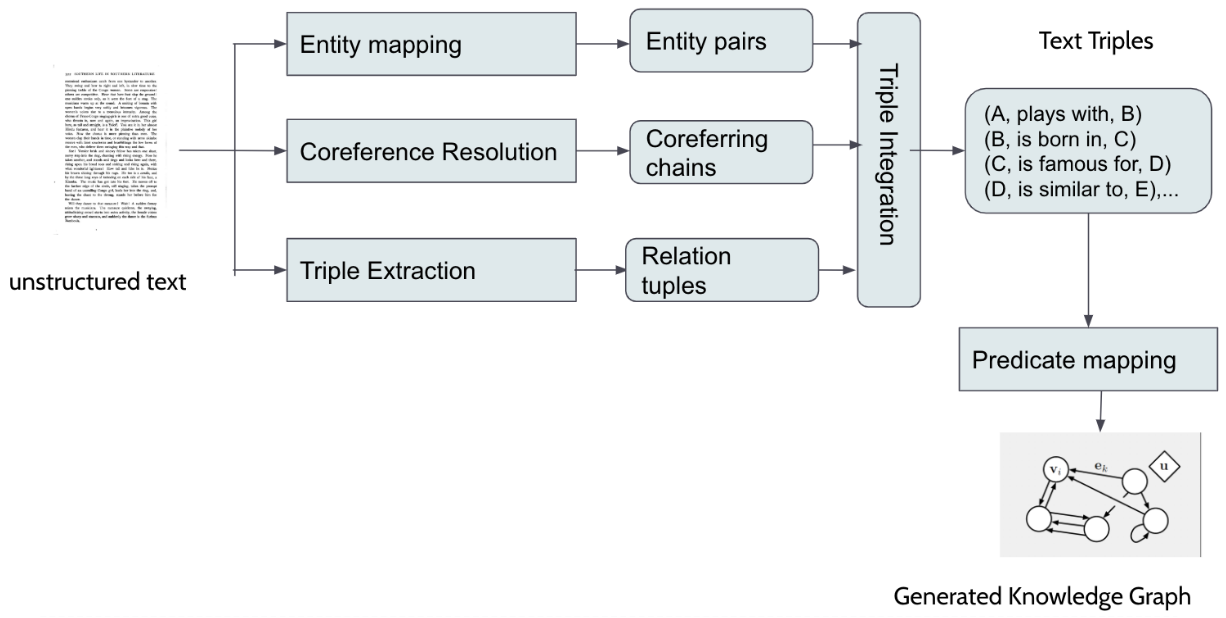


Figure 1. Our pipelined process of generating knowledge graphs from text.

2.2 Entity Mapping

The aim of the Entity Mapping component is to map an entity extracted from unstructured text to an ID as output. If an extracted entity can be mapped to an identical entity in the knowledge graph, the ID of such an entity in the KG would be used as representative for the extracted entity. Otherwise, a new ID is assigned to the extracted entity. The expected results of this component are a set of mapping entities for each extracted entity. This mapping also facilitates other tasks down the pipeline, as will be shown later.

2.3 Coreference Resolution

We then used Allennlp's source file to run coreference resolution extraction on the raw data text. This produced a cluster of nouns and pronouns that refer to the same entity. Each cluster was essentially an equivalence class that contained entities referring to the same thing. We for each equivalence class, we picked a “most representative entity” to represent the class by finding the most frequently used entity that wasn’t a pronoun. An alternative way to find this most representative entity would have been to find the first mention of the entity that wasn’t a pronoun. Both were valid options that produced similar results, but we went with the former method because it was simpler and more consistent.

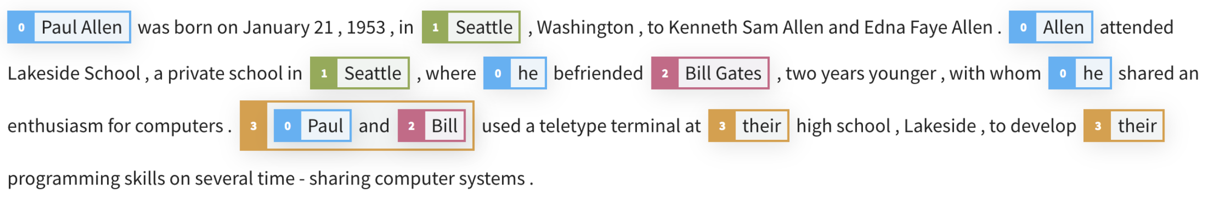

Paul Allen was born on January 21 , 1953 , in Seattle , Washington , to Kenneth Sam Allen and Edna Faye Allen . Allen attended Lakeside School , a private school in Seattle , where he befriended Bill Gates , two years younger , with whom he shared an enthusiasm for computers . Paul and Bill used a teletype terminal at their high school , Lakeside , to develop their programming skills on several time - sharing computer systems .

Figure 2. Example of Coreference Resolution Classification

2.4 Triple Extraction

The quality of the knowledge graph is largely dependent upon the extracted relation tuples, so we tried a couple of different existing Open Relation Extraction systems and also built our own system. We found that the Relation Extraction model from AllenNLP and our transformer model yielded the best results.

2.4.1 Triple Extraction using an existing open IE system (AllenNLP)

We used the OpenIE model from AllenNLP to find relationships in each sentence of the text, each of which contained one predicate and at least one entity. We then filtered by relationships that contained exactly two entities separated by a predicate since this related two entities by a direct relationship to generate our relation tuples.

We also noticed that many extracted entities were not proper objects (nouns), which could not be incorporated into a knowledge graph. Therefore, we validated the extracted entities using a part-of-speech

(POS) tagger and removed all tuples with entities starting with a verb/prep./etc.. We also truncated entities with long names. Examination over a large sample showed that this additional validation step was highly effective in filtering out malformed tuples.

2.4.2 Triple Extraction using a sequence-to-sequence model

Relation extraction is at the core of our pipelined process, yet none of the existing relation extraction models yields satisfying triples that can be directly used to generate a graph (without the need of any validation procedure). In particular, tuples generated by Stanford Open IE are overly abstract and miss many important details we might care about in the original sentence. AllenNLP Open IE keeps both a truncated and detailed extraction, but tend to mistakenly cast verbs or adjuncts as entities. Hence, we decided to develop our own relation extraction model and compare it with existing systems.

Inspired by Neural Machine Translation, we formulate the open relation extraction problem as a sequence to sequence generation task. More specifically, given an input sentence, the task is to produce a sequence containing the subject, object, and predicate delimited by special tokens indicating the type of each span. For instance, given the input "Barack Obama was the President of the United States." our objective is to generate the sequence "<arg1>Barack Obama </arg1> <rel>was</rel> <arg2>President of the United States </arg2>.

Given the recent success of self-attention-based neural networks in a variety of sequence processing works, we apply the transformer architecture (Vaswani et al., 2017) to this problem. A transformer is composed of an encoder and a decoder, each of which contains several blocks. An encoder block contains a multihead attention layer followed by a fully connected feed-forward layer. The decoder block looks the same except the first multi-head attention layer is masked so that it can not peek ahead at future inputs, and it has an additional multi-head attention layer that gets some of its inputs from the corresponding encoder block. The input sentences are first tokenized using a byte-pair tokenizer from OpenAI's GPT model (Radford et al. 2018). The final output of the last decoder block is fed into a final linear layer with a softmax activation function to generate a probability distribution over the output vocabulary.

In addition to using the tokenizer from OpenAI's GPT, we also used the pretrained token embeddings to warm start our models as these embeddings were the result of a significant amount of training on much larger corpora (so they might capture some low-level features of the language). The size of our source and target vocabulary is about 35000 (tokens). Because of memory limits, the training set was split into chunks and each chunk was run for 1-2 epochs. We used the Adam optimizer with a decaying learning rate schedule. We also applied dropout to linear layers with 0.1 drop probability. Finally, we employed label smoothing with epsilon = 0.1 to make the model more unsure, a technique that had been shown to improve accuracy. Our final model has dimension 768, feed-forward dimension 1200, 6 self-attention heads, and 5 encoder and decoder blocks. In total, the model has 123,506,387 parameters.

We trained the model over 4 days on a single Nvidia K80 with 12 GB of vRAM. We had a batch size of 9000 tokens, and batches were sorted by sentence length to minimize the number of padding tokens to maximize efficiency. We used training data[2] made available by (Cui et al., 2018), which was gathered from Wikipedia dump 20180101. The whole training set contains 36,247,584 (sentence, tuple) pairs in total.

[2]https://1drv.ms/u/s!ApPZx_TWwibImHl49ZBwxOU0ktHv

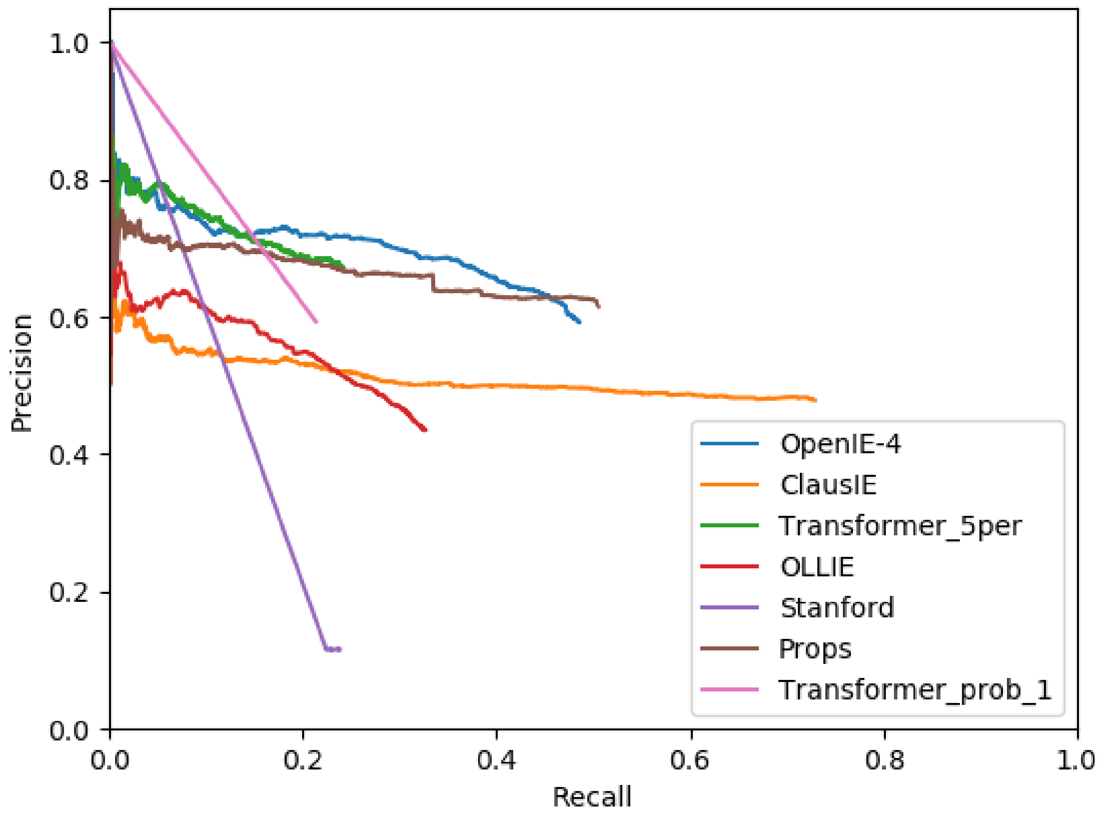


Figure 3. Precision-Recall Curve of extractions by our Transformer model and 5 other Open IE systems

We also evaluated our model on an Open IE benchmark[3] created by (Stanovsky et al., 2018), with 3200 sentences from Wikipedia and the Wall Street Journal that has 10,359 extractions. For each sentence, our extraction was lined up with multiple gold extractions and the extraction was considered valid if it achieved a high enough lexical coverage of the reference (defined by the percentage of words covered). Subsequently, the precision and recall of our system were analyzed on different confidence thresholds, and the area under the PR curve calculated.

[3] https://github.com/gabrielStanovsky/oie-benchmark

It is observed from the Precision-Recall curve (Figure 3) that our transformer model performs significantly better than 4 of the 5 existing rule-based Open IE systems and achieves comparable results with the current best systems OpenIE-4. None of our models achieves a recall as high as some other systems though (noticeably ClausIE, which is best at recall) because our model only produces one extraction per sentence.

2.5 Triple Integration

The relation tuples produced decent results, but often contained many ambiguous entities, like pronouns. For example, we might have the following cluster of entities that refer to the same person: [‘Obama’, ‘he’, ‘Barack’ ‘Barack Obama’, ‘Barack Obama’, ‘president of the United States’]. We refer to these clusters as equivalence classes because all the entities are equivalent in terms of what they represent. To resolve these entities, as well as collapse nodes that really referred to the same thing when building the graph, we used the equivalence classes generated in the coreference clustering. For each entity, we replaced it with its **most representative entity (MRE)** from its respective equivalence class if it had one. We picked either the most frequent or first occurrence (excluding pronouns) entity as the most representative entity, and in practice we found that the first method worked slightly better.

Many entities were made up of nouns or pronouns that were found in multiple clusters. For these cases, we only replaced the entity if at least half the entities were in the same equivalence class. This helped resolve problems where a complex entity made up of multiple smaller entities was replaced by a single entity from the group.

2.6 Predicate Mapping

To generate a knowledge graph, it was necessary to combine equivalent/similar relations. As an example, relations such as “premiered” and “released” as pertaining to a movie should be considered the same relation. To do this, we first mapped the relations to a vector space using the TransE knowledge graph embedding. TransE works by representing the entities (**h**, **t**) as vectors in a d-dimensional space with the relation vector (**r**) being a translation vector between the two. Mathematically, this can be described as optimizing the scoring function f_r (h, t) = - $\| \mathbf{h} + \mathbf{r} - \mathbf{t} \|_{½}$ under the constraints $\| \mathbf{h} \|_2 = \| \mathbf{t} \|_2 = 1$. This was trained on the triples that were generated. Once the embeddings were completed, the relation vectors were combined if their cosine similarity score was above a certain threshold.

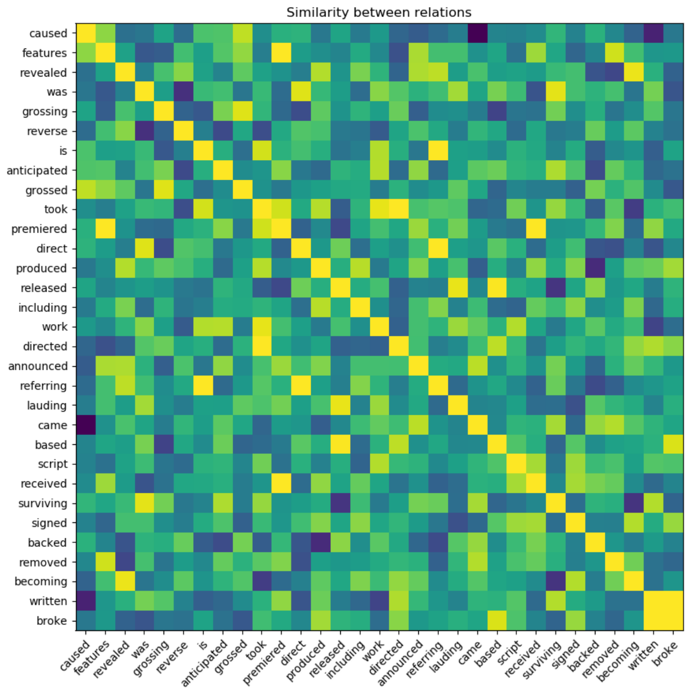


Figure 4. Heatmap demonstrating similarity between various relations

3 Results

3.1 Tuples

Our outputs are open relation tuples of the form (subject, relation, object). Below are some examples.

Input sentence	**Extraction by our transformer model**
As a group , the team was enshrined into the Basketball Hall of Fame in 1959 .	the team ||| was enshrined ||| into the basketball hall of fame in 1959 .
Certain fractional quantum Hall phases appear to have the right properties for building a topological quantum computer .	certain fractional quantum hall phases ||| to have ||| the right properties for building a topological quantum computer .
Ballast tanks are equipped to change a ship ’s trim and modify its stability .	Ballast tanks ||| are equipped ||| to change a ship ’s trim and modify its stability .
In Taiwan , the locals speak a version of the Minnan language which is called Taiwanese .	the minnan language ||| is called ||| taiwanese .
In 2004 it was expected that redevelopment work in the remaining subway would probably obliterate what remains exist .	redevelopment work in the remaining subway ||| would probably obliterate ||| what remains exist .
The town and surrounding villages were hit by two moderate earthquakes within ten years .	the town and surrounding villages ||| were hit ||| by two moderate earthquakes within ten years .
There were 22.2 % of families and 23.8 % of the population living below the poverty line , including 15.8 % of under eighteens and 37.5 % of those over 64 .	22.2 % of families and 23.8 % of the population ||| living ||| below the poverty line .
These are known as Porter ’s three generic strategies and can be applied to any size or form of business .	Porter ’s three generic strategies ||| can be applied ||| to any size or form of business .

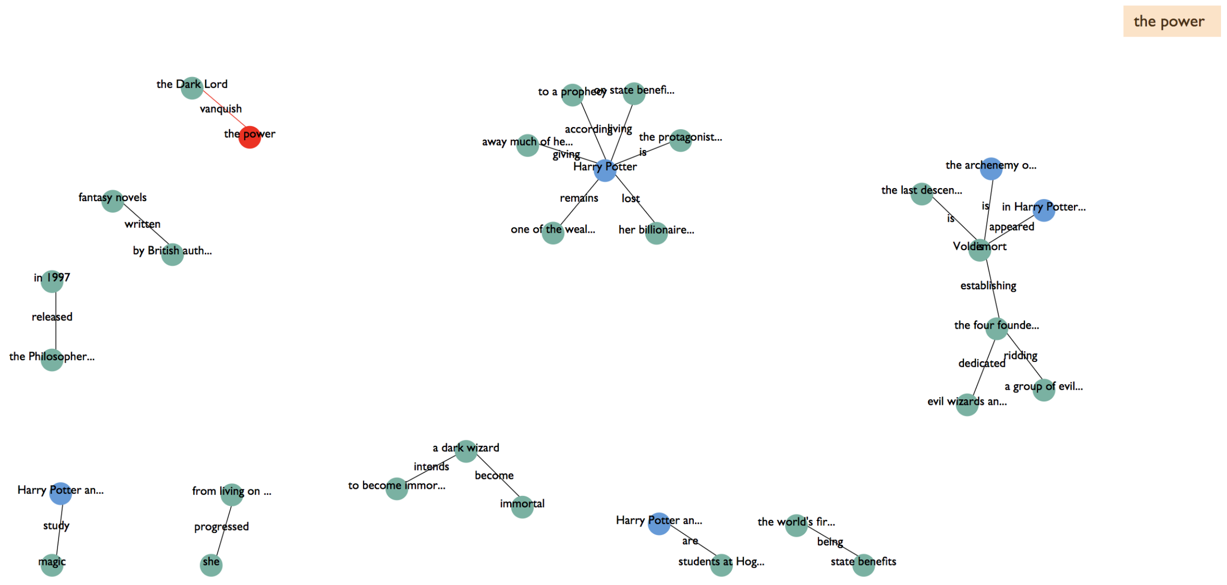
Table 1. Sample open relation extractions by our transformer model

We observed that our transformer is able to correctly identify the boundary of arguments and relations in most cases. Sometimes our transformer is even able to correctly replace an ambiguous pronoun in an argument with what it's referring to in the context, as in the last example above. But eventually, all the pronouns would be replaced during coreference resolution and triple integration.

3.2 Graph Visualization

We decided to use a force-directed graph to visualize relation triples and how they are laid out in a knowledge graph. The force layout is a class of graph layout algorithms in D3 that calculates the positions of each node by simulating an attractive force between each pair of linked nodes, as well as a repulsive force between the nodes. We assigned entities to each node and relations to the graph’s links / edges. Typically, the attractive force acts like a spring between the nodes, calculated by Hooke’s law. On the other hand, two nodes are pushed away from each other using Coulomb’s law. It is a commonly implemented graph drawing algorithm because of its flexibility and intuitiveness, as it requires no special knowledge of graph theory. We also decided to use the force layout for its interactivity and customizability.

In the beginning, we tried using other techniques for visualizing knowledge graphs. In particular, we experimented using the networkD3 package in R and ShinyR for creating interactive web apps and visualizations. However, after a few weeks of experimenting with various features, we decided that using D3 would allow us to customize our graph more freely.



In general, this viz does a decent job at showing two things. First, it shows the relationships between entities and lays it out in an organized manner. Secondly, we added features that allow us to drag nodes, and be able to see individual relationships (highlighted in red) when hovering over them as well. This gives the viewer an engaging interactivity aspect.

That being said, there are improvements that could be made to the visualization. A method of better aligning entities would reduce visual clutter, and panel of some sort could be added on the side that prints out the full relational sentences when hovering over a node or edge.

4 Future Goals

Improving Recall (in relation extraction)

Despite the superior accuracy of our transformer system, our main weakness is with recall. Because we can only make 1 extraction per sentence, our approach is inherently limited on a data set with over 10000 triples and 3200 sentences. One potential workaround of this limitation, keeping within our general approach, is to use something akin to a modified beam search decoding, basically a parallel hypothesis greedy search. The modification would be to start the search with an initial set of candidates that already contain the first token of candidate arguments. The search would then decode each candidate into its most likely triple. Each fully decoded candidate would be associated with a probability value as before. This value could be used as a threshold, and then we can take the top K candidates that achieve this threshold.

Downstream Application

Our initial motivation for building a knowledge graph is to help with the task of question-answering. Our entity and predicate mapping processes greatly reduced the heterogeneity problem and increased searchability over a knowledge graph. As a result, we can see its potential in question answering and other query expansion tasks.

More Interactive Visualization

As mentioned before, we would like to find a way to make the text and overall graph more organized. Additionally, we would like to implement more interactive features for the user to explore relations.