The Study of Knowledge Graphs and their potential application in Firebase

CSC 400 Fall 2022

Independent Study Final Report (Completed 12/4/2022)

Author: Jerry Huo

Advisor: Franz J. Kurfess

Cal Poly San Luis Obispo, Computer Science and Software Engineering Department

Abstract

This project aims to study knowledge graphs - to understand how they work, how they may be improved, and how they may be potentially applied to Firebase. The reason is to further my own knowledge regarding knowledge graphs as well as to the potential benefit and optimization of data for the AI for Search & Rescue research team. Research will be conducted on knowledge graphs, and then a knowledge graph will be developed using the "Wildland Search and Rescue Missions by NYS Forest Rangers beginning 2012" database (Data.ny.gov). Finally, research will be done to determine if knowledge graphs can be used in Firebase.

Introduction

After discovering how knowledge graphs are a big component to Google's query functionality, I was amazed by the power and potential they hold. Obviously, as seen with their widespread use, their power has been uncovered by many. I hope to do the same, and understand the nuances behind the details of knowledge graphs. Furthermore, the AI for Search and Rescue project utilizes Firebase as its data storage. If knowledge graphs could be utilized in Firebase to organize all the data, the results would be tremendous.

Background

My personal knowledge graph will be built using Python and Google Colab. I chose to use Google Colab over any other platform due to its convenience. I don't have all the libraries I need to develop my knowledge graph downloaded onto my computer, and I don't need to with Google Colab. The libraries that are used are the following: re, pandas, rdflib, BeautifulSoup, spacy, urllib, networkx, and matplotlib.

Related Works

There are numerous applications of knowledge graphs throughout the web. The most notable one would be Google. The usage of Google's knowledge graph can be seen on a panel to the right of the results page after you submit a query. That panel contains the most relevant and important information that resulted from your query. Some of the other more notable knowledge graphs include but are not limited to the following: Cyc, NELL, DBpedia, ConceptNet, GDelt, and so much more (Dery, 2016). Some of their purposes are the following: Cyc is a knowledge graph developed to be able to imitate a human's common sense, NELL was developed to learn to read the web, and DBpedia extracts information from Wikipedia. There are so many knowledge graphs out there because there are so many different uses for knowledge graphs.

Research

Knowledge is situational, layered, and changing. As such, data is hard to master and organize as the context behind the knowledge is lacking in traditional methods. Knowledge graphs are built for this, and are a type of a semantic graph that works to convert internet data into knowledge comprehensible by machines - representing the relationships and facts about concepts. At its core are sets of nodes and edges, where nodes are the facts and edges are the relationships between nodes. These nodes are further organized using semantic triples. Semantic triples allow us to make statements about resources with the structure <Subject> <Predicate> <Object>. The subject and object are the two resources being related, and the predicate is the relationship. For example: <I> <am a> <student> (Manola, 2014).

There are many ways to develop a knowledge graph, and 3 of the most common techniques are manual curation, creation from (semi) structured sources, and creation from

unstructured sources (Heist, 2020). Manual curation is providing data manually. Creation from (semi) structured sources is more efficient, and takes data from the web. Creation from unstructured sources is harder than from (semi) structured sources, but is better in that it has a lot more data since it extracts knowledge from free text. Overall, manual curation is the worst since it is the hardest technique to simplify, and the other two have their own advantages and disadvantages.

"Knowledge graphs can help with, but not limited to, data governance, fraud detection, knowledge management, search, chatbot, recommendation, as well as intelligence systems across different organizational units" (Pan). This is due to the fact that knowledge graphs provide a lot of flexibility through data unification. The power that knowledge graphs hold is only limited to the scope of the data that is available to it. With more data, more analysis can be done and more inferences can be made. As such, queries are completed easily. Since the relationships between data have already been established, the context behind knowledge is present.

Although there are some disadvantages to knowledge graphs, there aren't too many. For example, manually curated knowledge graphs are incredibly time consuming and expensive. The cost of development for *Cyc*, one of the oldest knowledge graphs, is estimated to be around 120 million dollars (Heist, 2020). Of course, the benefits of Cyc far outweigh the drawbacks that came from its initial development. Cyc is powerful in that its ontology spans "all" human concepts and its inference engine is able make the same inferences and come to the same conclusions that a human would.

Firebase is a NoSQL database program built on Google's infrastructure that is a backend service which provides developers various tools and services to help them develop apps and structures JSON data. Firebase structures its data as a JSON tree, with individual data added as a

node and can be located by a key (Firebase). Knowledge graphs can be developed with JSON and does seem pretty promising (Futia, 2022. The issue comes from being able to add it into or use it with Firebase. Firebase suggests developing a model using TensorFlow Lite and then integrating it into the database. Besides that, it doesn't offer any other solution.

System Design / Implementation

My knowledge graph will be a simpler one, and will be developed using manual curation, with an excel database and will have a semantic triple consisting of incident number, response type, and found (true/false). For example: <Incident #> <rescue> <true> or <Incident #> <search> <false>. The goal for my knowledge graph is to be able to organize and display the graph of the data given as well as transform this data into personalized and componentized content to have a query function.

```
[18] EG = Namespace("")

     def create_eg_uri(name : str) -> URIRef:
         quoted = quote(name)
         return EG[quoted]

     incident_mapping = {
         "uri": "INCIDENT NUMBER",
         "INCIDENT NUMBER": create_eg_uri("incidentNum"),
         "RESPONSE TYPE": create_eg_uri("responseType"),
         "FOUND IN SEARCH AREA": create_eg_uri("found"),
     }
```

Figure 1

The above picture (Figure 1) shows the process of url-encoding these values and the uri references which will end up being used for queries. It's also creating an ontology that shows the relationship between the incident and the response and whether or not the response was successful.

```
query = """SELECT ?incident ?responseType
WHERE {
  ?incident eg:found ?responseType .
}"""
results = graph.query(query, initNs={"eg": EG})
for row in results:
    print(row)
```

Figure 2

The above picture shows the usage of the uri references to complete a query, identifying whether the incident resulted in finding the missing persons, or not.

```
(rdflib.term.URIRef('NY-2018-WS-286'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-284'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-283'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-282'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-281'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-279'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-278'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-277'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-276'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-275'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-274'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-273'), rdflib.term.Literal('True'))
(rdflib.term.URIRef('NY-2018-WS-352'), rdflib.term.Literal('False'))
(rdflib.term.URIRef('NY-2018-WS-345'), rdflib.term.Literal('False'))
(rdflib.term.URIRef('NY-2018-WS-338'), rdflib.term.Literal('False'))
(rdflib.term.URIRef('NY-2018-WS-337'), rdflib.term.Literal('False'))
(rdflib.term.URIRef('NY-2018-WS-336'), rdflib.term.Literal('False'))
(rdflib.term.URIRef('NY-2018-WS-334'), rdflib.term.Literal('False'))
(rdflib.term.URIRef('NY-2018-WS-327'), rdflib.term.Literal('False'))
(rdflib.term.URIRef('NY-2018-WS-324'), rdflib.term.Literal('False'))
(rdflib.term.URIRef('NY-2018-WS-318'), rdflib.term.Literal('False'))
```

Figure 3

Figure 3 shows some of the results returned by the query. The screenshot is taken at the point where “True” stops and “False” begins in the results.

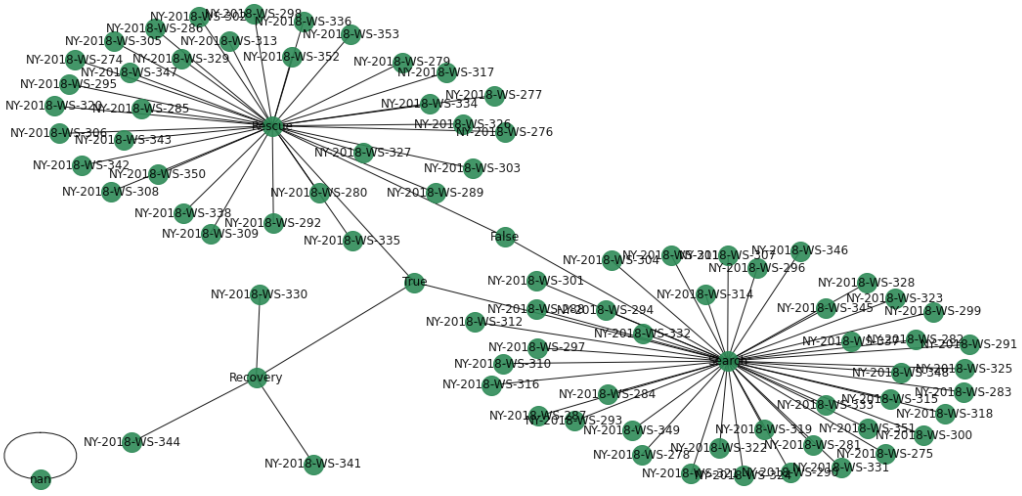

Figure 4

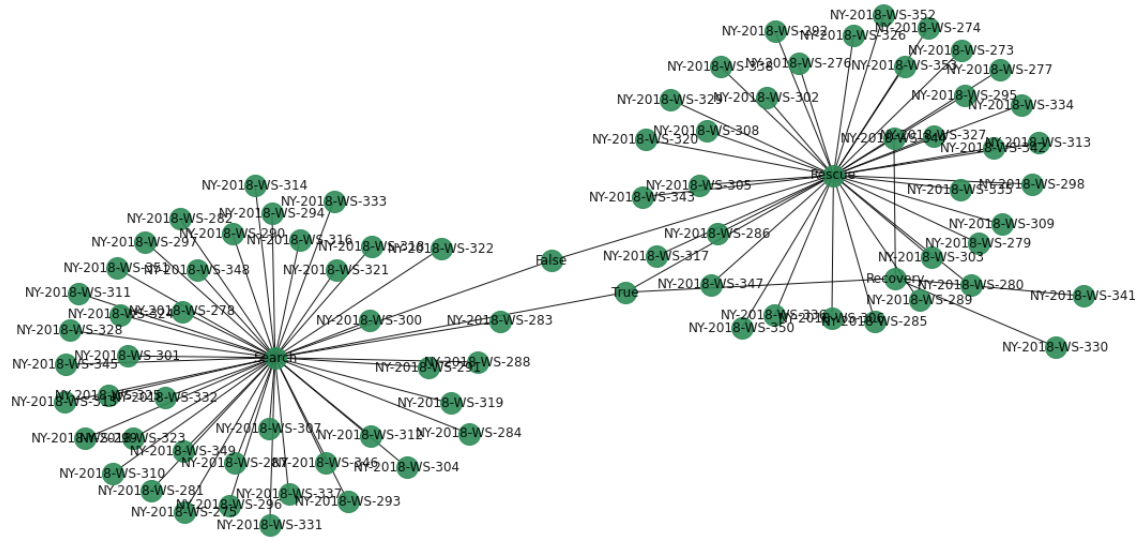

Figure 5

Figure 4 and 5 are the visual representations of the knowledge graph. Figure 5 is the graph in which the data is limited (only has the incident number, response type, and found in search area. Figure 4 is the graph in which the data is not limited.

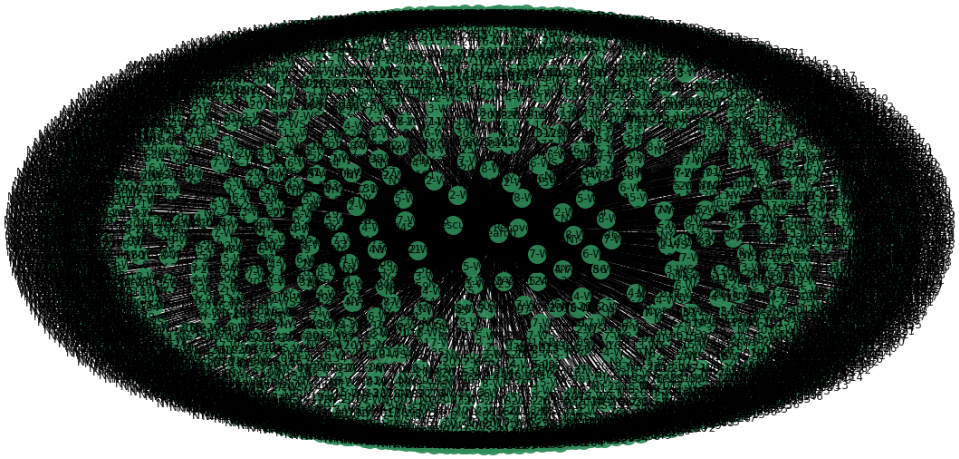

Figure 6

Figure 6 is not limited in any way, and is quite difficult to make an interpretation of any kind due to the sheer number of incidents there are in the database.

Testing

The knowledge graph will be tested using the database "Wildland Search and Rescue Missions by NYS Forest Rangers beginning 2012". The database is fairly large with 3235 incidents, so only the first 100 incidents will be used for development and visual purposes. Besides the number of incidents, the database will first be limited to incident number, response type, and found in search area. After the knowledge graph is deemed working, then the database limits will be removed.

Future Work

I hope to be able to improve on my knowledge graph in all aspects. Of course, one of the biggest improvements I intend on pursuing is the implementation of a web scraping functionality, and just making my knowledge graph more flexible overall. Further improvements can be made by creating it from (semi) structured sources. There's also using the knowledge graphs that are available to all, such as OpenCyc. Although OpenCyc is no longer supported by Cycorp, there are still multiple copies available on the web. Although further research is required, in terms of how OpenCyc might be useful, if its inference engine is able to come to the same conclusions as a human, then the AI for Search and Rescue team may be able to use it to predict where a person in need of rescue may have chosen to go.

Conclusion

The scale and power of knowledge graphs are a lot greater than I originally imagined. Every knowledge graph developed has a lot of utility and plays a major role, and creating a knowledge graph on my own was very fun and educational. At the beginning, there were a lot of challenges, the main ones being what database to use, how I wanted to develop it, and what data to organize. Surprisingly and thankfully, there is a vast ocean of wealth available to help explain and answer my questions. At the current point of the world's research into knowledge graphs, there isn't too much to improve on, and right now the focus is on how to use knowledge graphs most effectively. Moreover, this research also led me to discover the potential of OpenCyc in regards to the AI for Search and Rescue team. OpenCyc may be able to predict the actions of missing persons, and if this possibility is true, that would be amazing.

Unfortunately, implementing knowledge graphs into Firebase seems to have limited potential. As a platform, Firebase is an amazing document-based system to store all the data for AI for Search and Rescue. Of course, it's too bad that Firebase itself doesn't provide much direction in terms of what can be done and TensorFlow Lite doesn't seem to be very flexible in terms of being able to implement a knowledge graph.

References

Data.ny.gov. "Wildland Search and Rescue Missions by NYS Forest Rangers beginning 2012". *Data.ny.gov*, https://data.ny.gov/Public-Safety/Wildland-Search-and-Rescue-Missions-by-NYS-Forest-/u6hu-h7p5/data. Retrieved 23 November 2022

Dery, Sebastien. “Challenges of Knowledge Graphs.” *Medium*, Medium, 21 Dec. 2016, https://medium.com/@sderymail/challenges-of-knowledge-graph-part-1-d9ffe9e35214. Retrieved 25 November 2022

Firebase. “Structure Your Database | Firebase Realtime Database.” *Google*, Google, https://firebase.google.com/docs/database/web/structure-data. Last updated 15 November 2022. Retrieved 25 November 2022

Futia, Giuseppe. “Building Knowledge Graphs from Structured Sources.” *Medium*, Towards Data Science, 23 Jan. 2022, https://towardsdatascience.com/building-knowledge-graphs-from-structured-sources-346c56c9d40e. Retrieved 26 November 2022

Heist, Nicolas, et al. “Knowledge Graphs on the Web - an Overview”. Germany. Data and Web Science Group. 12 March 2020. Retrieved 17 September 2022

Kornev, Daniel. “Knowledge Graphs in End-User Products: From Cyc to Ai Assistants - Part I.” *LinkedIn*, 29 Apr. 2020, https://www.linkedin.com/pulse/knowledge-graphs-end-user-products-from-cyc-ai-part-daniel-kornev/. Retrieved 26 November 2022

Manola, Frank, et al. *RDF 1.1 Primer*, https://www.w3.org/TR/rdf11-primer/#section-triple. 24 June 2014. Retrieved 25 November 2022

Pan, Jeff, et al. “Knowledge Graphs.” *The Alan Turing Institute*, https://www.turing.ac.uk/research/interest-groups/knowledge-graphs#:~:text=Knowledge%

20graphs%20can%20help%20with,systems%20across%20different%20organisational%20units. Retrieved 23 November 2022

Stardog Union. “What Is a Knowledge Graph: Stardog.” *Stardog Union*, https://www.stardog.com/knowledge-graph/. Retrieved 25 November 2022