



# Fall 2022 Capstone Project
# Final Report

# Generating Knowledge Graphs from Unstructured Financial Data

## December 17, 2022

**Team Mentors**
Mr. Satish Banka
Mr. Rajkumar

**Team Members**
Alex Kita (ak4729)
Arnav Saxena (as6456)
Hyo Won (Elin) Kim (hk3175)
Ridwan Olawin (ro2372)
Shashwat Singh (ss6373)

# Introduction

## Background

This capstone project is being sponsored by Accenture, an IT professional services company. The ultimate aim of this project is to build a pipeline capable of generating knowledge graphs from unstructured financial data sources such as earning calls, quarterly reports, news articles etc and then building a natural language querying system to easily fetch information from this knowledge graph.

With the advent of semantic web, the ability to fetch interlinked documents holds and still holds great value. Knowledge Graphs can be considered as the next step in the evolution of semantic web where instead of interlinking the documents, the knowledge and content that is contained within these documents is interlinked. With ever increasing data volume this becomes even more important for efficient information extraction tasks. Before diving into the problem statement and motivation behind this project, this section gives background information of important concepts used in this report.

**Important jargons used in this report:**

**Knowledge Graph:** Knowledge graph is a knowledge base that uses graph-like structure to store and link data. It is a directed graph explaining semantics.

**Triple**: A knowledge graph is essentially a collection of triples of the following form: <*subject, predicate, object*> or <*subject entity, relation, object entity*>. In simple words, given a sentence, a triple would contain entities (node) and a relation (edge) that gives the relationship between them. For eg. in the sentence “Barack Obama was born in Hawaii” the triple that can be extracted would be <*Barack Obama, born_in, Hawaii*>

**Ontology:** An ontology is a semantic data model of a knowledge graph that is generated by clustering triple entities and relations into broader classes often arranged in hierarchical order. For instance, in the same example as above, entity “Barack Obama” can belong to the ontology class person or US president or anything else depending upon the domain

**SPARQL:** SPARQL is a query language that is used to fetch information from knowledge graphs and other linked data sources

## Problem Definition

Knowledge based industries such as consulting and finance are heavily dependent on extracting information from unstructured financial documents such as powerpoint presentations, financial statements, news articles, website content etc for generating value. However, they mostly rely on manually parsing through this data to achieve the

goal which clearly is inefficient given the ever increasing volume of this data. To deal with this problem, our team worked on building a knowledge graph based question answering system. The idea is to enable efficient information extraction such that professionals can easily query this knowledge graph in natural language and get fast answers, ultimately aiding and improving their decision making process . For the scope of this project we have limited ourselves to the cryptocurrency domain.

**Existing Work**

The process of generating a knowledge graph and querying system can be broken down into three major components - triple generation, ontology building, and natural language to SPARQL query conversion system.

Triple generation methods can be broken down into unsupervised and supervised methods. A baseline unsupervised method for triple generation involves parsing a dependency tree then extracting subject, predicate, and object using heuristic rules. An example of the same is ClauseIE [1]. In our work, however, we have focussed on experimenting with supervised systems owing to better performance showcased by them. The models we explore are REBEL by Bablescope [2], KnowGL[3] by IBM, PURE [4] by Princeton University, and Allen NLP's SRL model [5]. REBEL, KnowGL, and AllenNLP perform the task of entity and relation extraction jointly. The way they differ is that REBEL and KnowGL use an autoregressive seq2seq model based on BART whereas AllenNLP uses a Semantic Role Labeling model built on top of BERT. PURE on the other hand claimed that better accuracy can be achieved by using different models for extracting entities and relations.

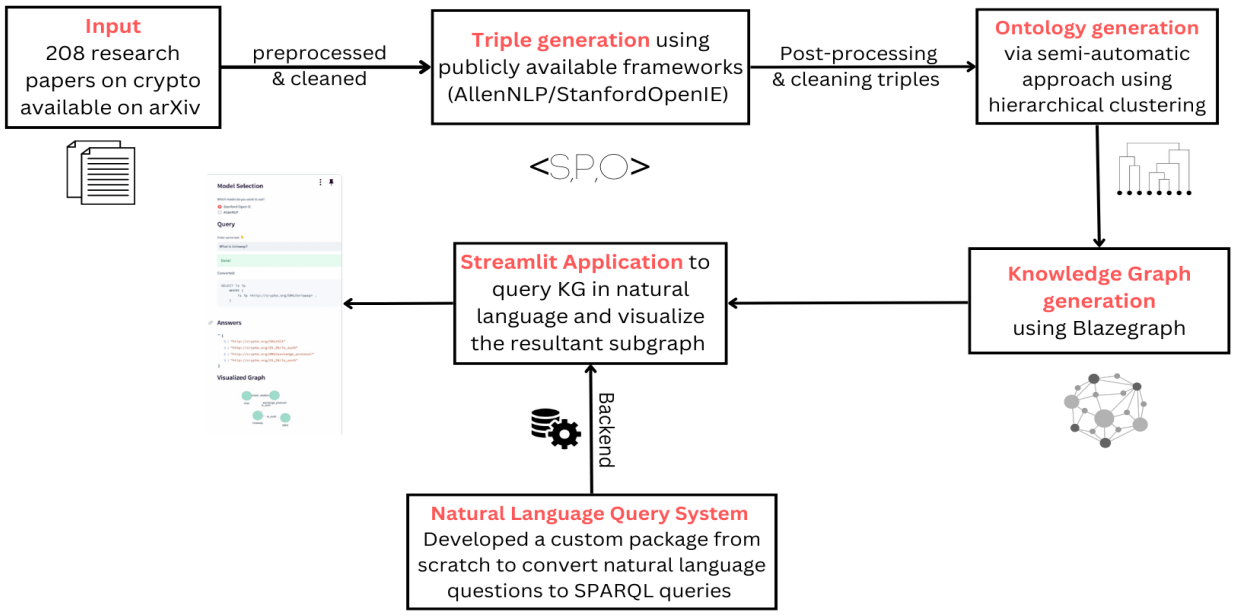
Previously ontologies have been explored as an Agent-based approach [6] or using data mining tools [7]. In agent-based approaches they used Syntactic Parsing followed by Extraction of concepts and extraction of relations which finally gave them the desired ontology. The data mining methods included Apriori algorithms with a combination of frequency of words in the corpus to give ontology. In Automatic Ontology generation techniques [8], there has been a lot of work done on morphologic transformation and solving ambiguity in the corpus. In our approach we are using a combination of all these approaches to build a semi-automatic ontology generation technique.

As per available literature, there are two broad types of Natural language to SPARQL query conversion systems available. The first type breaks down the system into three broad steps - template classification, query triple generation and mapping, and a query construction module. The second type performs the three actions as a joint task effectively using an architecture similar to that of machine translation models ([9]). In this work we have focussed on the first type simply because the second type of models

require large amounts of custom labeled training data that we didn't have. Works such as Queepy[10] and AquaLog[11] are ontology independent query triple generation systems that use dependency parsing and analysis to extract query triples. We use a similar approach as was also discussed in [12] and have pretty much re-implemented that paper with a couple of tweaks specifically in the query triple mapping step.

### Overall Approach

The flowchart below describes our approach in brief.




As suggested by our mentors initially we decided to scrape research papers from arXiv belonging to the cryptocurrency domain. The task henceforth was to build a knowledge graph and query system with these papers as our domain. Given the enormous scope we divided the task into 5 steps. As the first step we extract triples from these papers using publicly available open source frameworks namely **Allen NLP and Stanford OpenIE**. Next, we build the ontology using a semi-automatic approach where-in we cluster the extracted entities and relations using hierarchical clustering. The triples and ontology are passed into **Blazegraph** to generate the knowledge graph.

To query this knowledge graph a custom natural language question to SPARQL query conversion system was built. To this end the three component system discussed in [12] was implemented from scratch. This system ultimately served as the backend of the Streamlit application which provided an easy to use UI to query our knowledge graphs in natural language and visualize subgraphs and query results.

# Methods

## Data

### Dataset

One of our interests in this project is to obtain useful information from *unstructured* documents within the financial domain. Though there are various financial areas to explore and types of documents available online, we decided to choose the arXiv research paper dataset [18] and specifically in cryptocurrency subdomain for the following reasons. 1) Research papers are unstructured as they are given as PDF documents. Typically, we use either manual processing or rule-based methods to identify headers, text, figures etc. in the documents. However, it has limitations in terms of the size of the dataset or the quality of extracted information. Therefore, we aim to tackle this technological difficulty with a machine learning model discussed below. 2) leveraging arXiv dataset leads us to generate useful knowledge graphs of the cryptocurrency domain. Cryptocurrency space keeps growing, and it is hard to catch up with all the new emerging concepts. Our knowledge graph built on the arXiv research papers enables users to get big pictures of complex cryptocurrency concepts without looking at papers. We extracted abstract and conclusion sections of 208 cryptocurrency papers from the arXiv dataset. For future work, we can extend it to a large dataset as we prioritized generating a complete end-to-end pipeline in this project.

### Pre-processing

As we analyzed research papers, there exists context-dependent sentences such as model results and figures. Since the dataset is given in PDF format, we have to denoise or remove information that heavily relies on the context. Hence, we adopted a machine learning model called GROBID for information extraction. GROBID is a machine learning library for extracting, parsing and re-structuring raw documents such as PDF into structured XML/TEI encoded documents with a particular focus on technical and scientific publications [17]. In this project, GROBID helps us recognize structures within documents such as sections, figures, descriptions, text etc. With this tool, we converted the PDF documents to JSON data and extracted the abstract and conclusion sections since the essential and non-context dependent information of the research papers are mainly in these sections.

### Database

We have used BlazeGraph [14] TripleStore to to save the generated tripes along with the ontology. BlazeGraph is an open source and a high performance graph database

which supports RDF and SPARQL APIs. It also provides a Python Client which makes the usage of this tool more easier to integrate with our project.

# Analytics Methods

## Model Exploration

After research and literature review, four models were explored.

| Models | Avg Precision | Avg Recall | F1 score |
|---|---|---|---|
| KnowGL | 0.18 | 0.07 | 0.1 |
| REBEL | 0.29 | 0.21 | 0.22 |
| Stanford OpenIE | 0.29 | 0.29 | 0.25 |
| AllenNLP | 0.55 | 0.5 | 0.49 |

According to the results displayed in Table 1, we finally selected the outputs of AllenNLP and StanfordOpenIE for Ontology generation and building Knowledge Graphs. Therefore, we have two Knowledge Graphs, one associated with the outputs from StanfordOpenIE and another one associated with the outputs of AllenNLP.

How we measured the accuracy metrics is shared in the appendix. Also, the details of REBEL and KnowGL are included in the appendix.

- **Stanford Open IE**: It is an open source, open domain information extraction software developed by the Stanford Natural Language Query Group and based on the paper [21]. It works by building a classifier that learns to extract self-contained clauses from longer sentences. Natural logic inference is then run over these short clauses to determine the maximally specific arguments for each candidate triple.This allows the system to have a greater awareness of the context of each extraction, and to provide informative triples to downstream applications. The following are sample outputs from the model.

  ***subject:*** *biggest individual insider shareholder* ***; relation:*** *is* ***; object:*** *Arthur Levinson*
  ***subject:*** *Levinson* ***; relation:*** *owns* ***; object:*** *4.5 million shares of Apple stock*
  ***subject:*** *Levinson* ***; relation:*** *owns* ***; object:*** *4.5 million shares*
  ***subject:*** *company* ***; relation:*** *according* ***; object:*** *most recent proxy filing*

  The triplets can be inferred from the above output with the subject, relation and object. In addition, the core difference between this software and others is that the schema for the relation labels does not need to be specified in advance.

- **AllenNLP Semantic Role Labeling model**: It is an open-source BERT based pre-trained model developed by Allen Institute for AI [16]. The model performs Semantic Role Labeling; in other words, it assigns semantic roles, such as verb, arguments, and supportive arguments (temporal, cause, etc.). The sample output is included in the Triplet Generation section.

### Hyperparameter tuning to increase output quality

- **Stanford Open IE**: There are a number of flags you can set to tweak the behavior of the program. Coreference resolution was selected to resolve pronouns to their canonical antecedent. The maximum number of entailment per clause was limited to 3 ensuring that entailments produced for each clause in a sentence did not potentially include redundant relationships. Strict triplets were also enforced ensuring that only logically warranted triples are extracted but it puts more burden on the entailment system to find minimal phrases

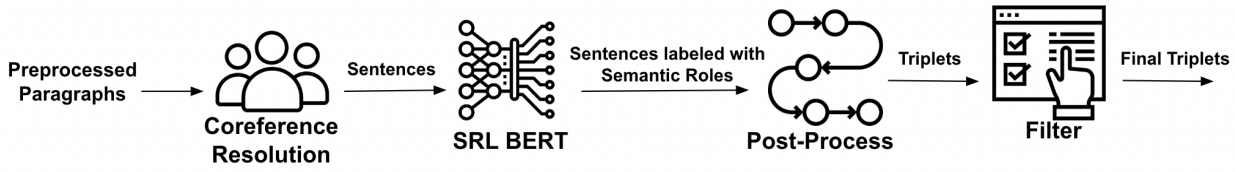
- **AllenNLP Semantic Role Labeling model**: There are no hyperparameters available to tune for this model. However, additional steps were performed for post-process to increase the quality of output. The details of the post-process are shared in the Triplet Generation section.

## Triplet Generation

Generating triplets is the foundation of this project, and this section explains the process of converting input sentences to triplet following sequential steps. Triplets are generated using two different approaches - one approach with the AllenNLP SRL BERT model and the other with the Stanford OpenIE model.

### AllenNLP

After data preprocessing, the input texts are passed through the following pipeline, which is composed of four steps, to generate the triplets.




First, coreference resolution is the task of finding all expressions that refer to the same entities in the text. The following shows an example of identifying the real-world entities of the pronouns in the text.

Before coreference resolution:

```
In other words , they have more income than they currently want to
spend ; they would like to save some of their money and spend it in
future time periods.
```

After:

```
In other words , the suppliers of funds have more income than the
suppliers of funds currently want to spend ; the suppliers of funds
would like to save some of the suppliers of funds's money and spend
some of their money in future time periods.
```

Second, after the coreference resolutions, the paragraphs are broken down into sentences and fed into the SRL BERT model. The model first uses WordPiece to tokenize the sentence, and the resulting sequences are fed into BERT to obtain contextual representation. Utilizing the context, the model labels semantic roles for relevant tokens. The sample output is included below to demonstrate this. Note that ARG1, V, and ARG3 represent semantic roles given the context of the sentence. The extensive list of semantic roles labels and explanation are included in Appendix.

```
[ARG1: IR] involves elements of communication , marketing , and
finance and is [V: designed] [ARG3: to control the flow of
information from the management of a public corporation to its
investors and stakeholders].
```

In the context or frame of the verb “designed,” the word “IR” plays the role of ARG1 (patient) and the phrase “to control… and stakeholders” plays the role of ARG3 (ending point). However, these labels do not specify the subjects and objects of the sentence. Hence, we introduce an additional post-process to address that.

In the post-process, we utilize the semantic role labels to identify the subject, verb, object in the sentences which make up the triplets. Verbs are already identified in the model output. As for the subjects and objects, we can observe (from the example above) that the argument labels with lower numbers play the role of subjects and higher labels objects. This pattern is true for broader examples as well. Therefore, we convert the lower number labels to subjects and higher number labels to objects. Additionally, we remove the model outputs that do not contain any argument labels since those outputs cannot be converted to triplets. Namely, outputs that tagged only verbs and did not produce any other tags were removed because we cannot extract any triplet in those sentences. Example is as follows:

```
Cash [V: can] often be received a significant amount of time after
the initial transaction.
```

Finally, we apply additional filtering rules to select “clean” triplets. Due to the nature of SRL BERT model output, the length of subjects and objects can be too long as shown in the example below:

```
Subject: IR
Verb: designed
Object: to control the flow of information from the management of a
public corporation to its investors and stakeholders
```

Note that O is lengthy, and this can be problematic to fit in the knowledge graph node. We want the entities (subjects and objects) to be short and concise so that they can be visualized as graphs where nodes represent entities and edges represent relation between nodes. For the filtering rules, we referenced the KnowText paper [15]. The details of the rules are listed below in the Stanford OpenIE section. In addition, we remove the sentences that are similar to each other based on the cosine similarity score of 0.5.

In summary, the original number of triplets that were output from SRL BERT + post-process is 9909. After the filtering step, the resulting number of triplets is reduced to 420. We acknowledge that this is a significant drop, potentially risking the loss of knowledge of the input text. However, we believe that these triplets still capture the core of the knowledge - the ontology section demonstrates that the classes of relations (verb) and entities (subject and verb) from AllenNLP are well-representing the appropriate relations and entities of the cryptocurrency domain.

### Stanford OpenIE

The exact same process for extracting triplets from the financial input sentences using StanfordOpenIE is similar to the AllenNLP process. Instead of using the SRL BERT system, coreferenced resolved sentences are passed to Stanford’s OpenIE system and then post-processed. The same filters in the post-processing of triples for both our triplet generating systems are the same.

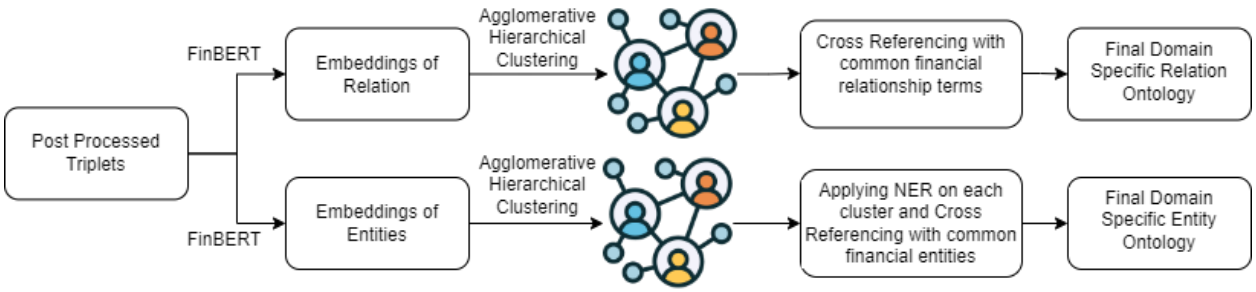
Below table is the outline of the post processing applied on both triplet extraction systems. These filters are effective showing a significant decrease in the number of triplets and increasing their accuracy.[15]

| **FILTER** | **DESCRIPTION** |
|---|---|
| max_word_count | S,P and O should be made of max 3 words |
| min_char_count | S,P and O should be made up of min 2 characters |

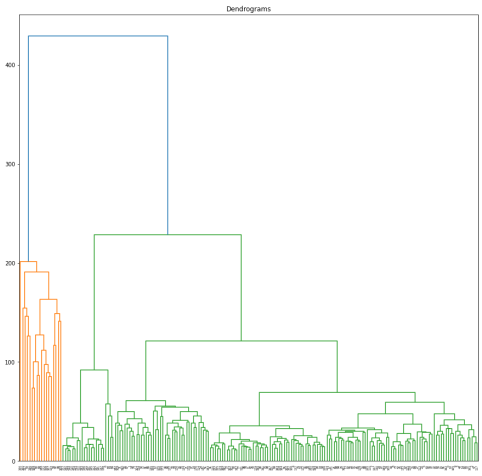
| | |
|---|---|
| day_of_week | Triples should not contain days of the week |
| nn_subject | S should be a noun |
| pronouns | Triples should not contain pronouns |
| duplicate_subject_word | S should not contain duplicate words |
| duplicate_object_word | O should not contain duplicate words |
| subject_verbs | S should not contain verb phrases |
| object_verbs | O should not contain verb phrases |
| subset_phrases_subject | When 2 triples exist with similar S, longer S is chosen |
| subset_phrases_predicate | When 2 triples exist with similar P, longer S is chosen |
| subset_phrases_object | When 2 triples exist with similar O, longer S is chosen |
| same_subject_object | S and O should not be the same |

## Ontology

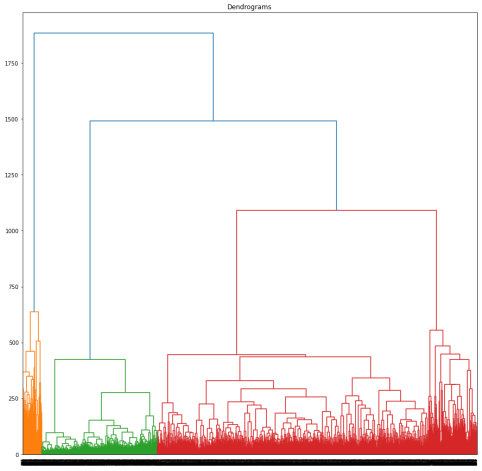
We generated both the relation ontology and entity ontology in a semi-automatic format which can be used to develop an ontology for different domains given different key terms. We used triplets extracted from the models as the input and passed them through FinBert Embeddings to generate embeddings for every unique relation and entity. We then applied Agglomerative Hierarchical Clustering on these embeddings to identify and group similar relations/entities. We also cross-referenced the clusters with common domain-specific financial relationships and entities along with NER to get domain-specific ontology. The flowchart for this process is shown below.




While generating FinBERT embeddings we took the average of the embedding in case the relation or entities had multiple words. Additional pre-trained FinBERT embeddings had an added advantage that they have been already trained on a large corpus of financial documents which makes embeddings more representative of the word in context.




This is the dendrogram generated from the agglomerative clustering on relations. After exploring the quality of each cluster we finally chose the number of relation clusters 20. After cross referencing these clusters with the domain specific vocabulary we got 19 unique types of relations which can be mapped to all the relations for outputs generated from AllenNLP. Similarly for StanfordOpenIE we chose 50 relation clusters and got 43 unique types of relations which were mapped to all the relations generated by the model.




This dendrogram is generated by the entities when agglomerative clustering was applied on them. We selected 25 clusters and 30 clusters for AllenNLP and StandfordOpenIE respectively. Further we applied Spacy NER on each of these clusters and got the max frequency of the named entity occurring in each cluster. We then cross referenced these named entities with domain specific entity vocabulary which finally resulted in 8 unique types of entities for AllenNLP and 14 unique types of entities for StanfordOpenIE.

## Results from Ontology Generation

- **AllenNLP**
  - Relations : ['UNDERLIE', 'IS AUTHOR OF', 'ENDORSE', 'REVERT', 'UNDERESTIMATE', 'PRECEDE', 'PREVAIL', 'CORRELATE', 'RECALCULATE', 'INVEST', 'CHARACTERIZE', 'OPTIMIZE', 'DECOMPOSE', 'INCUR', 'TRANSFER', 'ANALYSE', 'COLLUDE', 'REGULARIZE', 'OUTPERFORM']
  - Entities : ['PORTFOLIO','MISCELLANEOUS','PERSON','ORG','BLOCKCHAIN','MINING','ATM','CONTRACT']

- **StanfordOpenIE**

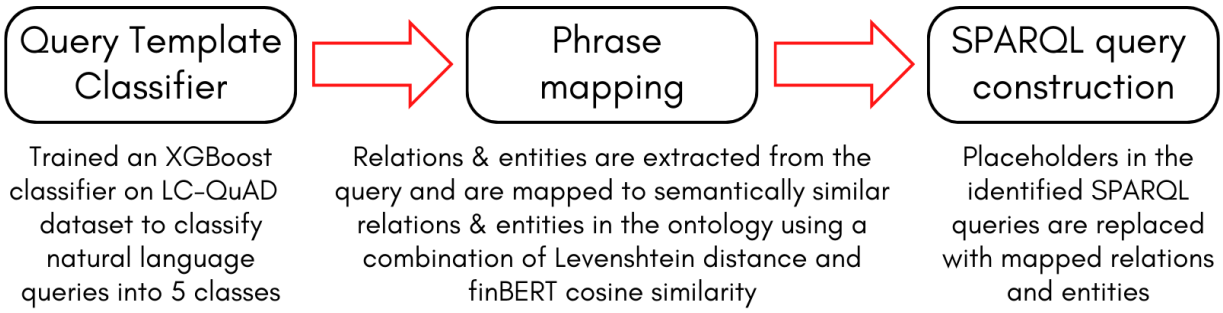
- Relations : ['YIELD', 'PLAY ROLE IN', 'LAG', 'ALSO MOTIVATE', 'PRECEDE', 'CATEGORIZE', 'ENDORSE', 'BE PROPORTIONAL TO', 'THEREBY ADVANCE', 'DIAGNOSE', 'ACCORD TO', 'NUMBER OF', 'HOWEVER REQUIRE', 'HAVE FOR', 'DECOMPOSE', 'REMINISCE', 'IS OVER', 'RECONSTRUCT', 'ASGARI', 'IS IN', 'PRICE', 'TRADE', 'DEBUNK', 'LASTLY INTRODUCE', 'SIGNIFICANTLY SIMPLIFY', 'ANTICIPATE', 'PERTURB', 'TYPICALLY BECOME IN', 'SELL', 'CRYPTO', 'EVEN JUMP OF', 'UNDERESTIMATE', 'DEPEND ON', 'INVEST', 'ENTAIL', 'MAY HAVE', 'DEEM', 'DATUM', 'CORRESPOND TO', 'ALSO FIND', 'PREVAIL', 'SIGNIFICANTLY OUTPERFORM', 'CHARACTERIZE']
- Entities : ['DERIVATIVE','LEGAL','CARDINAL','MISCELLANEOUS','PORTFOLIO','PERSON','CRYPTOGRAPHY','DEFI','ORG','FIAT','BLOCKCHAIN','ATM','CONTRACT','DATE']

## Knowledge Graph Generation

After getting the results of both the triples and the ontology of those triples from both triplet generating systems (Stanford OpenIE and AllenNLP), we developed an automatically generated knowledge graph by mapping the ontology of each S,P and O text and creating a RDF (resource description framework) based format for the mapped triples. The formatted triples were then loaded as a graph, serialized into a .n3 file, then loaded into BlazeGraph. This allowed us to simply pass a query (sparql) into BlazeGraph whenever a stakeholder needs to retrieve some information and get triples results based on the query.

## Natural Language Questions to SPARQL Query conversion

The real value of any knowledge graph (or any knowledge base for that matter) emerges only when it is easily queryable. To this end we developed ***text2sparql*** - a natural language question to SPARQL query conversion system specific to our knowledge graph. For the development of this module we primarily implemented [9] from scratch and broke down our system into three components. The figure below summarizes the same.




**Component 1: Building a query template classifier**

An XGBoost template classifier was built that classified any natural language question into 5 SPARQL query templates. These five templates along with example questions and queries are listed with examples in Table 2.

LCQuAD dataset [13] which has 5000 natural language queries classified into 38 SPARQL templates was used to train the model. Since not all the 38 templates are equally covered in the dataset, we filtered out ~2500 sentences that covered the 5 templates (~500 samples per class). After standard text pre-processing of queries as has been covered in previous sections, we used pretrained finBERT model for generating 786 dimensional embeddings for each question. These embeddings served as inputs for the model while the template labels served as the output. Please note that as suggested by [12] we did experiment with fasttext embeddings but the model performance on using finBERT embeddings was much higher. Further a train-validation-test split of 80:10:10 was done for hyperparameter tuning. The final trained model had the following parameters:

```
n_estimators=400, max_depth=35, learning_rate=0.01,
eval_metric='logloss'
```

This model achieved ~82% accuracy on the LCQuAD test set. Further testing was done on 25 manually generated questions specific to our domain and the classification accuracy on those questions stood at 77%.

**Component 2: Phrase Extraction & Mapping**

This component in-turn had two steps.

**A. Query triple extraction:**

**For relation extraction, rule-based pattern matching** was used to extract predicate phrases from the query. These patterns essentially extract verb phrases and/or “dobj” and its identifiers from a query. The following patterns were used.

```
{'POS': 'VERB', 'OP': '?'}
{'POS': ADP, 'OP': '*'}
{DEP: {"IN": ["amod", "compound"]}, 'OP': '*'}
{'DEP': 'dobj', 'OP': '+'}
```

For **entity extraction noun phrases were extracted** using Spacy-roBERTa's noun chunking model. After extracting these predicate phrases and noun phrases we performed standard text cleaning on them. The phrases were lemmatized and any adjective, adverb, or auxiliary verbs were removed. This step had an efficacy rate of more than 90%.

**B. Phrase mapping:** Finally, a combination of Levenshtein distance and finBERT cosine similarity was calculated between these cleaned entity and relation phrases and finalized entities and relations in our triples respectively. Ultimately we obtain the closest entity and relation present in our ontology which is semantically and morphologically closest to our verb and noun phrases. Only those entities or relations are mapped where in the Levenshtein distance and finBERT similarity score is more than 0.75. After obtaining the mapped entities and relations, their classes can be easily obtained by performing a simple lookup in the ontology

**Component 3: SPARQL Query Generation**

After obtaining the query template from step 1 and query triples and their ontology classes from step 2, we generate all possible combinations of queries by replacing these entities and relations in the template placeholders. Any query for which a triple doesn't exist in our triple list is filtered out while the remaining queries are outputted as the final result. In most cases this module will return 1 query per question unless there are multiple valid entities and relations are present.

Table 2 provides some examples of different queries constructed by our model for sample natural language questions.

| Template number | Type | Explanation | Query Template | Example question | Example query constructed |
|---|---|---|---|---|---|
| 1 | List (type 1) | Given S find all relations | `SELECT DISTINCT ?P ?O WHERE { <S> ?<P> ?O.}` | Who is Ramit Sawhney? | `SELECT DISTINCT ?P ?O WHERE { <http://crypto.org/PERSON/Ramit_Sawhney> ?<P> ?O.}` |

| | | | | | |
|---|---|---|---|---|---|
| 2 | List (type 2) | Given S,P, find all Os (conversely given all Os and Ps find all S) | `SELECT DISTINCT ?O WHERE { <S> <P> ?O.}` | What papers did Ramit Sawhney write? | `SELECT DISTINCT ?O WHERE { <http://crypto.org/Person/Ramit_Sawhney> ?<http://crypto.org/Profession/Is_Author_Of> ?O.}` |
| 3 | Count | Given S/O and P find count of all such pairs | `SELECT (COUNT(?O) as ?COUNT) WHERE {<S> <P> ?O.}` | How many papers did Ramit Sawhney write? | `SELECT (COUNT(?o) as ?COUNT) WHERE {<http://crypto.org/Person/Ramit_Sawhney> <http://crypto.org/Profession/Is_Author_Of> ?o.}` |
| 4 | Aggregation (Min) | Given S/O and P find the minimum O/S | `SELECT ?O WHERE { <S> <P> ?O. } ORDER BY AESC(?O) LIMIT 1` | Which paper by Ramit Sawhney has the lowest citations? | `SELECT ?O WHERE { <http://crypto.org/Person/Ramit_Sawhney> <http://crypto.org/Aggregate/Citations> ?O. } ORDER BY AESC(?O) LIMIT 1` |
| 5 | Aggregation (Max) | Given S/O and P find the maximum O/S | `SELECT ?O WHERE { <S> <P> ?O. } ORDER BY DESC(?O) LIMIT 1` | Which paper by Ramit Sawhney has the highest citations? | `SELECT ?O WHERE { <http://crypto.org/Person/Ramit_Sawhney> <http://crypto.org/Aggregate/Citations> ?O. } ORDER BY DESC(?O) LIMIT 1` |

**Table 2**

This table showcases the working of our natural language question to SPARQL query conversion system *text2sparql.* "S" = subject, "P" = predicate, and "O" = object present in the query triple. "Example query constructed" are the actual final queries as generated by text2sparql for questions asked in the "Example question" column

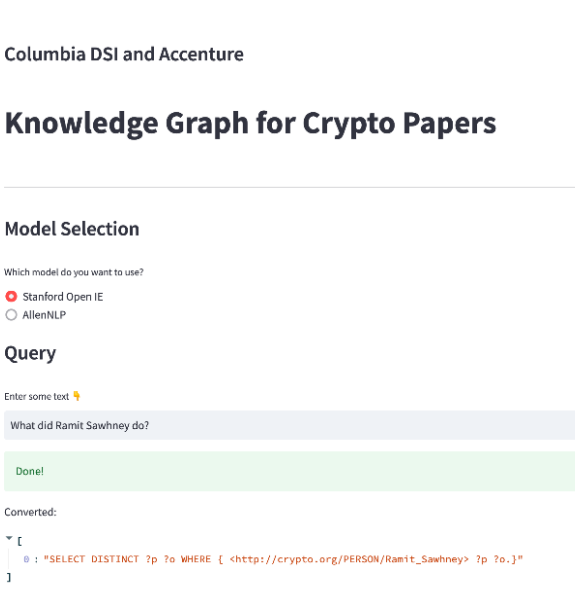
## Evaluation and Results

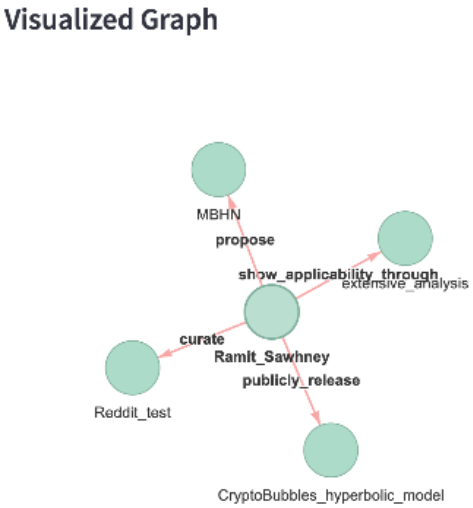
After doing a lot of research, we discovered that there is no SOTA way to evaluate how good a knowledge graph is and the quality of its results. Hence, we decided to manually choose 100 random sentences from the financial text that follows our template generation pattern when converting from natural language to SPARQL and binarily classify whether the knowledge graph outputs the results we want. The accuracy of the above approach resulted in 7% accuracy for StanfordOpenIE and 11% accuracy for AllenNLP. While these accuracies represent holistic accuracy, the table below shows component-wise accuracy for both models:

| Component-wise Accuracy | AllenNLP | Stanford OpenIE |
|---|---|---|
| F-1 Score Before Filtering | 0.49 | 0.25 |
| Query Conversion Acc. | 75% | 75% |

# App

We also developed an app where users can query on our knowledge graph without having any technical knowledge. This application offers several functionalities. First, it allows users to input a natural language query and convert it into SPARQL queries. Based on the generated SPARQL queries, the app shows the query results. Behind the scenes, the app interacts with our Blazegraph database through a SPARQL client called Pymantic [19]. Pymantic is a semantic web and RDF library for Python where you can load a n3 file into the graph database and query data with SPARQL. On top of that, we leverage our ontology and visualize a knowledge graph in which users can easily understand the relationships among cryptocurrency concepts. For visualization, we utilized streamlit-agraph library [20] that allows us to visualize a graph given triplets on streamlit. Although some of the functions are broken, we developed our own visualization methods inspired by their Triple Store implementation. As such, we successfully implemented a graph visualization system with an end-to-end pipeline on the streamlit. This interface looks like the following images.



# Discussion

## Conclusion & Future Work

We designed the 5-stage workflow to introduce structure through a knowledge graph. We collected 208 research papers (abstract and conclusion) focused on cryptocurrency and extracted the texts that contain essential information from the PDF documents. Then, we pre-processed and passed the texts through two different models, AllenNLP and Stanford OpenIE to generate triplets. The F-1 score accuracies for the models were 0.49 and 0.25, respectively. For ontology, we utilized the unique relations and entities from extracted triplets and performed agglomerative clustering along with cross referencing with domain specific relation/entities for semi-automatic ontology generation. For the SPARQL query conversion system, we primarily implemented work done in [9] from scratch. The accuracy for the conversion was 75%. Finally, we implemented an interface where users can input questions and receive query results in return.

There are some of the improvements which can be done on the existing project. The quality of triplets generated from the triplets extraction models can be made better by exploring more post-processing techniques. Additionally graph embeddings can also be used right after triplet extraction to generate knowledge graphs. This process would help getting the best relevant triplets, structuring a better ontology and generating a knowledge graph. These can also help in an effective query system where we would just be traversing the graph to find the most optimal answers to the query.

## Ethical Considerations

This project does not raise ethical concerns.

## Individual Contribution

• **Alex Kita:** Completely owned research and information extraction from PDF research papers and application development of the end-to-end pipeline; main researcher and developer for finding datasets and reading papers of the SOTA models for End-to-End Relation Extractions such as REBEL, and conducting hyperparameter tuning and investigation for fine-tuning of pre-trained REBEL.

• **Arnav Saxena:** Completely owned research and development of the natural language to SPARQL query conversion system; contributed to understanding the working of all

triple generation models covered in the report and their error analysis; main contributor on finetuning of REBEL transformer model on custom dataset;

• **Elin Kim:** Completely owned the 4-step triplet generation process using the AllenNLP SRL BERT model; main researcher for maximizing the performance of SRL BERT; contributed to setting up local connection to blazegraph; organized the evaluation methods and metrics generation to measure the accuracy of our product.

• **Ridwan Olawin (team captain):** Main researcher and developer for setting up and maximizing the performance of Stanford OpenIE software for the purpose of this project; set up recurring team project meetings and liaising with third parties (professors, mentors) as needed. Main contributor on converting triples to RDF format and parsing it into BlazeGraph.

• **Shashwat Singh:** Completely owned the semi-automatic ontology generation technique; contributed towards ideation of knowledge graph generation; main researcher and developer for KnowGL implementation to generate triplets from the model; main developer for implementation of fine-tuning and re-training of REBEL model on custom dataset; research and literature review on automatic ontology building techniques; research on suitable graph databases to store triplets.

# References


[1] Luciano Del Corro, Rainer Gemulla; ClauseIE: Clause-Based Open Information Extraction System; International World Wide Web Conference 2013

[2] Huguet Cabot, Pere-Lluis and Navigli, Roberto; REBEL: Relation Extraction By End-to-end Language generation; Findings of the Association for Computational Linguistics: EMNLP 2021; pages 2370-2381; Association for Computational Linguistics.

[3] Nandana Mihindukulasooriya, Mike Sava, Gaetano Rossiello, Md. Faisal Mahbub Chowdhury, Irene Yachbes, Aditya Gidh, Jillian Duckwitz, Kovit Nisar, Michael Santos, and Alfio Gliozzol; Knowledge Graph Induction enabling Recommending and Trend Analysis; ISWC 2022; Computing Research Repository.

[4] Zhong, Zexuan and Chen, Danqi; A Frustratingly Easy Approach for Entity and Relation Extraction; North American Association for Computational Linguistics (NAACL);Association for Computational Linguistics.

[5] Peng Shi and Jimmy Lin; Simple BERT Models for Relation Extraction and Semantic Role Labeling; arXiv preprint arXiv:1904.05255 (2019).

[6] Agent-based approach for building ontology from text

[7] Automatic Building of an Ontology from a Corpus of Text Documents Using Data Mining Tools

[8] Automatic Ontology Generation Using Extended Search Keywords

[9] Xiaoyu Yin, Dagmar Gromann, and Sebastian Rudolph; Neural Machine Translating from Natural Language to SPARQL

[10] Queepy

[11] Vanessa Lopez, Michelle Pasin, Enrico Motta; AquaLog: : An Ontology-Portable Question Answering System for the Semantic Web; Elsevier (2005), pp. 546-562, 10.1007/11431053_37

[12] Shiqi Liang, Kurt Stockinger, Tarcisio Mendes de Farias, Maria Anisimova & Manuel Gil; Querying knowledge graphs in natural language; *J Big Data* 8, 3 (2021). https://doi.org/10.1186/s40537-020-00383-w

[13] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann; LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs;International Semantic Web Conference, pages 210--218, (2017)

[14] BlazeGraph

[15] Knowtext

[16] Peng Shi and Jimmy Lin; Simple BERT Models for Relation Extraction and Semantic Role Labeling

[17] GROBID

[18] ArXiv dataset

[19] Pymantic

[20] Streamlit-agraph

[21] Gabor Angeli, Melvin Johnson Premkumar, Christopher D. Manning; Leveraging Linguistic Structure For Open Domain Information Extraction; Association for Computational Linguistics; pages 344−354, (2015)

# Appendix

## Code

All our work above has been published in [our GitHub Repository](#).

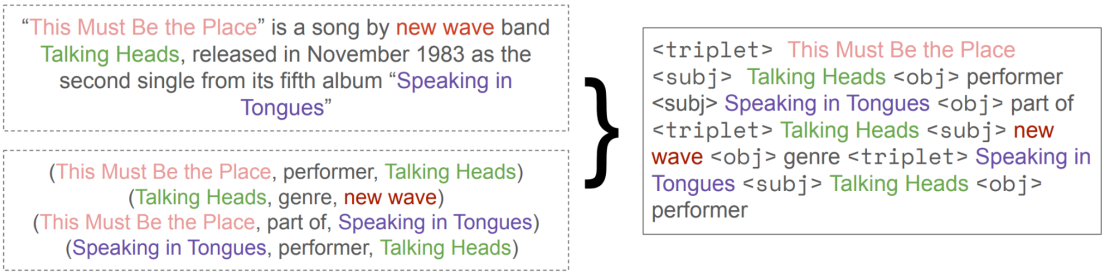
## Model Exploration

### Accuracy measurement - Ground Truth

To benchmark and better understand the pros and cons of each model we created a small test dataset of 10 short length documents with each document belonging to different financial domains so as to capture a variety of context. Triplets for each of these models were then generated manually which served our ground truth. These documents were then passed through the four models above to generate the model triplets. Precision, Recall, and F1 scores were calculated by manually trying to match the model generated triplets with the manually generated triplets. Exact matching wasn't used; instead if model generated triplets were semantically to manually generated triplets they accounted for a match.

### Details of models

- **REBEL**: It is a fine-tunable seq2seq model that performs end-to-end Relation Extraction (RE) that achieves the SOTA for many RE task datasets. It takes text as input and outputs "'linearized triplets", the sequence-version of triplets, which simplifies a RE task as a seq2seq task.




The base model is BART [4], a language model which combines BERT (bidirectional encoder) and GPT (autoregressive decoder) trained by reconstruction tasks of corrupting documents. REBEL leverages the pretrained BART-large model and fine-tunes it further with their own REBEL dataset that is mainly extracted from abstracts of Wikipedia.

- **KnowGL**: It is a transformer based generative Relation Extraction model and is primarily trained on the REBEL dataset. It was built by IBM as an internal tool to keep track and search of all the projects and the people who are working on them. They released a public version of their model on hugging-face on 15th September 2022. Additionally, they store their Knowledge Graph on BlazeGraph TripleStore and the Knowledge Graph is built on a weekly basis. Triplets generated follow this pattern

  *[(subject mention # subject label # subject type) | relation label | (object mention # object label # object type)]*

  Example Sentence : *Traders also kept a close eye on Tesla Inc., which pared gains substantially on news that Elon Musk is proposing to buy Twitter Inc. for the original offer price of $54.20 a share.*

  Output : *[(Tesla Inc.#Tesla, Inc.#enterprise)|chief executive officer|(Elon Musk#Elon Musk#human)]*

  Some of the advantages of this model is that it is available on hugging-face and performs decently on the domain data. But one the major drawback is KnowGL is not tunable since majority pre-processing is done by KnowGL parser which is not available in public domain. Additionally, there is no information about the model architecture or the parser so that it can be re-generated from scratch.

## Hyperparameter tuning

Each model went through intensive hyperparameter tuning as is discussed below:

- **REBEL**: Though REBEL is fine-tunable, we first conducted experiments with a pre-trained REBEL to capture the pros and cons of REBEL. Since it was pre-trained, the tunable hyperparameters are to change only decoding strategies. Precisely, we tuned the size of beam search, length penalty, the number of returned sequences, and the span length. The parameters that significantly affected the results were the beam size and length penalty which are useful to adjust how greedy the decoding would be and add a penalty for the shorter sequences in decoding respectively. The best yet computationally inexpensive model was with `(beam_size, length_penalty, num_sequences, span_length) = (5, 5, 1, 128)`.

- **KnowGL**: Since there is no public information about model architecture and there are no details about the parameters this model takes, hyperparameter tuning was not possible for KnowGL.

# AllenNLP SRL BERT - Semantic Role Labels

| Role(역할) | Definition(정의) |
|---|---|
| ARG0 | Agent |
| ARG1 | Patient |
| ARG2 | Start point/Benefactive |
| ARG3 | Ending point |
| ARGM-ADV | Adverbial |
| ARGM-CAU | Cause |
| ARGM-CND | Condition |
| ARGM-DIR | Direction |
| ARGM-DIS | Discourse |
| ARGM-EXT | Extent |
| ARGM-INS | Instrument |
| ARGM-LOC | Locative |
| ARGM-MNR | Manner |
| ARGM-NEG | Negation |
| ARGM-PRD | Predication |
| ARGM-PRP | Purpose |
| ARGM-TMP | Temporal |