

Drug Knowledge Graph Report

September 2021

Team Venkat, Mariyam, Harsha @ GENPACT

Overview

The following report summarizes learnings, observations and findings carried out by the team during the tenure of the Knowledge Graph Academy training. The tools used are Apache Ni-fi, Stardog and Protege. The latter half of the report consists in detail about the project - Drug Knowledge graph and the insights generated.

Goals

1. **Experimenting with the tools:** Try out different processors in Ni-fi, injecting data using Ni-fi into Stardog through SMS files, writing SPARQL queries to generate insights for the injected data, and learning how to construct an ontology on Protege.
2. **Mini Project:** Select a good dataset, inject it using Ni-fi into Stardog and frame questions to visualize the data and build an ontology for the same.

INTRODUCTION

Knowledge Graph

Originally known to have started as a project around 1980 which was kind of a semantic network, but with some added restrictions to facilitate algebraic operations on the graph. Tim Berners-Lee eventually coined the term semantic web in 2001 and by 2012 Google announced its Knowledge Graph and named it the same. While Knowledge Graph has existed as a concept for decades now, there is no one true definition to it. Every company/organisation has their

definition and function of the Knowledge Graph. In a nutshell, a Knowledge Graph essentially links data for easy, better and faster understanding.

Data Science has been advancing rapidly in the last few years and at this juncture, it is important to point out how Machine Learning and Knowledge Graphs together can improve the accuracy of systems and extend the range of machine learning capabilities. Machine learning models have been getting better at multiple tasks with great precision. However, some of the major issues faced in today's machine learning are data insufficiency, zero-shot learning, and explainability. Knowledge graphs complement the said problems by providing a more contextual approach to the predictions made and summarizing the decision-making process.

Most of the personalised content that we see on the web today is a result of combining both ML and KG. Moving further we will explore the tools used in this training and project.

TOOLS

Apache NIFI:

Apache NIFI will automate the flow of data between systems. The phrase 'flow of data or data flow' can mean different things in different contexts. In the case of Apache NiFi, it simply means the automated and managed flow of information between two or more systems.

Apache NiFi is mainly used for data acquisition, transportation, a guarantee of data delivery, capable of handling complicated and diverse data flows, inclusive of data-based event processing with buffering and prioritized queuing, and last but not least it is a user-friendly visual interface for development, configuration, and control which has many long term benefits.

Apache NiFi is a broad and thorough platform. It has potential for all systems and devices and can be used in a variety of ways. The users of this software will without a doubt experience the ease of data flow between systems. It can be overwhelming to set up and hence, Apache NiFi consulting services can be utilized to understand and apply this software.

Apache NIFI Features:

- **Web-based user interface**
 - It has a web-based user interface that is perfect for monitoring, designing, and control.

- **Configurable**
 - Easily configured to suit the different needs of different users. It can be customized to fit particular user requirements.
- **Security**
 - It has encrypted content, authorization, SSL, SSH, etc. which means it is safe and secure for users.
- **Data Provenance**
 - One can easily track data from beginning to end.
- **Extensions**
 - It will allow rapid development and quick testing.
- **Prioritized Queuing**
 - This will enable one or more prioritization options about data collection from queuing.

Stardog:

Today's data landscape is increasingly complex, varied, and distributed. The emergence of IoT, rise in unstructured data volume, and increasing frequency and complexity of analytics requests all contribute to the need for a more flexible data management solution. To keep up with enterprise demands for data and analytics, it's necessary to use tools that are designed to flexibly integrate data sources for varying purposes.

Stardog, the leading Enterprise Knowledge Graph platform, lets you connect data silos, no matter their structure, velocity, or location. Using a powerful, standards-based graph query language, SPARQL, Stardog can answer complex questions from data stored across multiple silos.

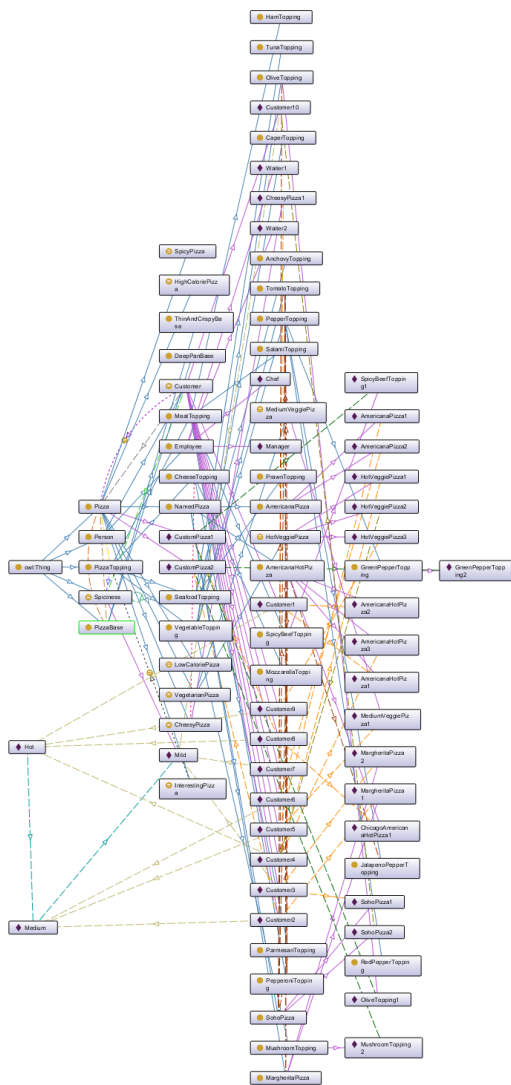
Ontology:

Defined as a set of concepts and categories in a subject area or domain that shows their properties and the relations between them; in simpler terms, one can define ontology as a blueprint of a dataset, constructed and used to understand the characteristics and relationships between the entities. In this training, we have used Protege for building ontologies.

Protégé is a free, open-source ontology editor and framework for building intelligent systems.

We started experimenting by building the famous pizza ontology first, where we explored different tabs and views in protege, the class hierarchy in entities, object properties, data properties, defined classes and more and finally using the Sparql and OntoGraf tabs for querying and visualising the developed ontology.

Further, we applied our learnings to develop a basic ontology for the chosen dataset which we will cover in the later topics.



PROJECT

About the dataset

Open Drug Knowledge Graph proposes a knowledge-based method to harmonize four heterogeneous sources into a single, comprehensive drug-centric knowledge graph.

1. **Wikidata** is one of the world's largest publicly accessible knowledge graphs, with over 90 million entities and over a billion statements. To get relevant data from Wikidata, look for drug (Q12140) entities that treat any condition and have any Drugbank ID (P715) (P2175). Additional elements include the active ingredient in the pharmaceutical (P3780), a major drug interaction (P769), and the ATCCode (P267). A total of 1,560 rows were collected from the query.

2. **Drugbank** is a drug-centric database with an emphasis on drug-drug interactions and bioinformatics. Its knowledge is delivered as an XML data dump. It has extracted all 2,166 medications from the XML dump from Drugbank, each with a maximum of 20 products and 100 interactions.

3. **WedMD** is a website dedicated to assisting people in their search for solutions for a specific ailment. The site provides an alphabetical index of all potential conditions. A list of medication therapies is provided for each illness. Because there is no public API for the website, it scraped its material programmatically. The crawler found 58,921 condition-drug relationships and 12,857 distinct medicines. Condition, product, user reviews, and prescription type are among the features extracted.

4. **GoodRx** is a healthcare company that keeps track of prescription medicine pricing in the US and offers free pharmacy coupons to help people save money on their meds. Because GoodRx does not offer a public API, they gather information on their drug goods straight from their website, starting with a Wikidata-based seed list. It extracted the following information from drug products: zip code, store, price type, price, and price link. For the 997 matched medication goods, a total of 20,688 prices and 23 shops were extracted.

JUSTIFYING THE CHOICE OF DATASET

As mentioned earlier, Knowledge Graph is still a developing field and there is a disparity in available public datasets which makes the selection of a good and functioning dataset a tedious task. However, we came across a recent work that was good enough to start with. The Open Drug Knowledge Graph dataset was curated by a couple of Grad students who wanted to simplify finding drugs and their prices through an app using ML.

The dataset is a result of web-scraping and finding relevant information. Some of the questions we wish to answer by the end of this project are the types of drugs used in products to treat conditions, some rare drugs used to treat just one condition, different stores the products are available at and justifying prices based on the current demand for the products. Healthcare is one of the most crucial and expensive businesses and it has been that way for a while. So we aim to figure out how the companies manufacturing the products using various drugs are playing the market.

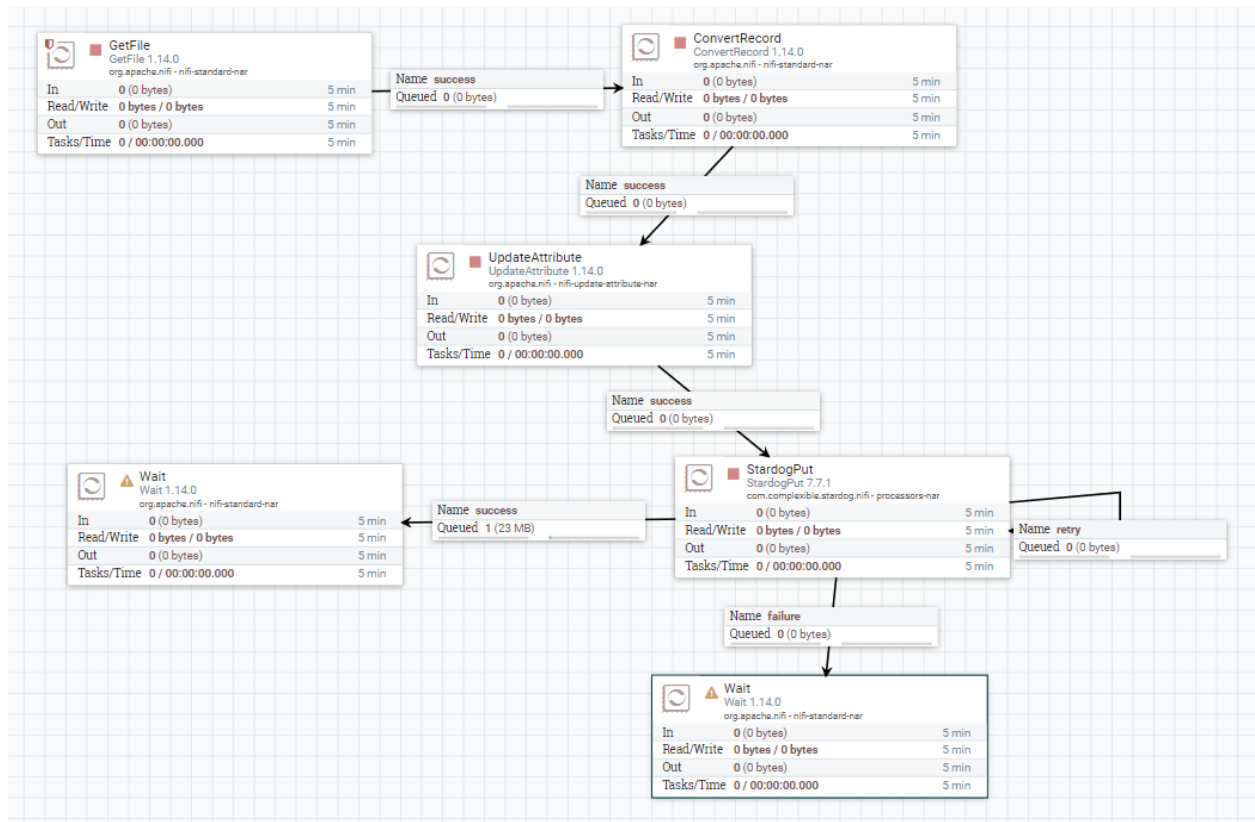
OVERVIEW OF THE STEPS TAKEN IN RESPECTIVE TOOLS:

Apache Nifi:

1. Start Nifi at localhost:8443
2. Create a process group named: OpenDrug in Apache Nifi
3. Download the Stardog NiFi nar files from <http://downloads.stardog.com/extras/stardog-extras-7.7.2.zip> into the `lib` folder in the NiFi installation folder
4. Add GetFile processor and set the input directory as the location of the folder containing the files.
5. Add “*.csv” to File Filter to import only CSV files from the folder.
6. Now add ConvertRecord processor to convert the CSV files to JSON format.
7. In ConvertRecord processor set Record Reader as CSVReader (if not present add the controller service).

8. Keep the default configuration of the CSVReader controller service and change the property value of **Treat First Line as Header** as True if it's false(It takes the first line/row of the CSV file as Header Line) and enable the service.
9. For the Record Writer choose JSONRecordSetWriter controller service, config the service to set Output Grouping value as One Line Per Object, keep rest as default and enable the service.
10. Add a processor called UpdateAttribute.
11. In the properties tab add a property called sms_file_name and set the value as `${filename:substringBeforeLast('.csv'):append('.sms')}`. It is basically used to map the SMS file w.r.t its CSV file name in the StarDogPut processor.
12. Add the StarDogPut processor and config the Client Service in the properties tab.
13. Choose Stardog controller service and config the Stardog connection string as the endpoint of your stardog, add the username and password of the stardog connection respectively and enable the service
14. In PutStardog set the following inputs for the properties :
 - Input format : JSON
 - Mapping file : `${absolute.path}/${sms_file_name}`
 - Base URI: `http://api.stardog.com/OpenDrug`
15. Now connect all the processors as follows and check the terminating relationship except success in the settings tab of each processor.

GetFile ---> ConvertRecord ---> UpdateAttribute ---> StardogPut



SMS File Structure:

PREFIX: <http://api.stardog.com/OpenDrug/Drugs/>

MAPPING

FROM JSON {

```

{
  "id" : "?id",
  "name" : "?name",
  "wiki_url" : "?wiki_url",
  "drugbank_url" : "?drugbank_url"
}

```



```

}
TO
{
?drugs a :Drug;

    :id ?id;

    :name ?name;

    :wiki_url ?wiki_url;

    :drugbank_url ?drugbank_url.
}
WHERE {

    BIND(template("http://api.stardog.com/OpenDrug/Drugs/id={id}") AS ?drugs)    }

```

- The structure of the **FROM JSON clause** closely resembles the source JSON structure with some changes
- The **TO clause** defines how the output RDF should look. This is the mapping target. It is analogous to the CONSTRUCT portion of the SPARQL CONSTRUCT query. It consists of a set of triples where variables can be used in any position.
- The **WHERE clause** is where you can transform source data and BIND the transformed values to new variables.
- **Construct IRIs** using the **TEMPLATE** function and **BIND** them to new variables.

Stardog & SPARQL

After injecting the data into stardog our major challenge was to check if the queries returned the right results. We cross verified the results with the dataset. Following are some of the questions we hoped to answer using the dataset

1. Find all products and drug components present in it

```
SELECT ?drug_name (GROUP_CONCAT(DISTINCT ?product_name;separator=", ")AS
?products)
```

```
WHERE{
```

```
    ?product a pdt:Product;
```

```
        pdt:name ?product_name;
```

```
        pdt:ref_drug_id ?refdid.
```

```
    ?refdid a drg:Drug;
```

```
        drg:name ?drug_name
```

```
}
```

```
GROUP BY ?drug_name
```

```
ORDER BY ?drug_name
```

```
Limit 100
```

<pre> 1 SELECT ?drug_name (GROUP_CONCAT(DISTINCT ?product_name;separator=", ")AS ?products) 2 WHERE{ 3 ?product a pdt:Product; 4 pdt:name ?product_name; 5 pdt:ref_drug_id ?refdid. 6 ?refdid a drg:Drug; 7 drg:name ?drug_name 8 } 9 GROUP BY ?drug_name 10 ORDER BY ?drug_name 11 Limit 100 </pre>	
Run to File	Text
Charts	Visualize
100 Results, 13506 ms	
drug_name	products
"(+)-menthol"	"Balsang Pain Relief"
"(-)-menthol 1-propylene glycol carbonate"	"Thera Derm Roll On"
"(12R)-12-hydroxyeicosapentaenoic acid"	"Luvira"
"(4-(trifluoromethoxy)phenyl)urea"	"Uremol 10%"
"(S)-camphor"	"Antiphilamine, Balsang Pain Relief, Doloran, Elixicure Lavender, Elixicure Original, Good Sense Ultra S"
"1,2-Distearoyllecithin"	"LumaSon"
"1,2-Docosahexanoyl-sn-glycero-3-phosphoserine"	"EnBrace HR, EnLyte, PramLyte"
"1,2-Hexanediol"	"Dr.G AQUASIS WATER VITAL SLEEPING MASK"
"1,2-icosapentoyl-sn-glycero-3-phosphoserine"	"EnBrace HR, EnLyte, PramLyte"
"1-(c14-c18 esteroyl)-2-docosahexanoyl-sn-glycero-3-phosphocholine"	"OB Complete Gold, Puralor"
"1-(c14-c18 esteroyl)-2-docosahexanoyl-sn-glycero-3-phosphoethanolamine"	"OB Complete Gold, Puralor"
"1-Palmitoyl-2-oleoyl-sn-glycero-3-(phospho-rac-(1-glycerol))"	"Surfaxin"
"13-cis-12-(3'-Carboxyphenyl)retinoic acid"	"Antiscar"
"2,2'-Dibenzothiazyl disulfide"	"T.R.U.E. Test Thin-Layer Rapid Use Patch Test"
"2-(l-menthoxy)ethanol"	"Pain Goodbye Aqueous Patch"

2. Find drugs for treating different conditions

```
SELECT ?drg_name (GROUP_CONCAT(DISTINCT ?cnd_name;SEPARATOR=", ") as ?conditions)
WHERE{
    ?treat a tts:Treatment;
        tts:id ?treat_id;
        tts:ref_condition_id ?ref_cnd_id;
        tts:ref_drug_id ?ref_drg_id.
    ?ref_cnd_id a cnd:Condition;
        cnd:name ?cnd_name.
    ?ref_drg_id a drg:Drug;
        drg:name ?drg_name.
}
GROUP BY ?drg_name
ORDER BY ?drg_name
```

<div> <div>Run</div> <div>Show Plan</div> <div>Reasoning</div> <div>opendrug</div> <div>Save to File</div> <div>Store Query</div> </div>	
<pre> 1 SELECT ?drg_name (GROUP_CONCAT(DISTINCT ?cnd_name;SEPARATOR=", ") as ?conditions) 2 WHERE{ 3 ?treat a tts:Treatment; 4 tts:id ?treat_id; 5 tts:ref_condition_id ?ref_cnd_id; 6 tts:ref_drug_id ?ref_drg_id. 7 ?ref_cnd_id a cnd:Condition; 8 cnd:name ?cnd_name. 9 ?ref_drg_id a drg:Drug; 10 drg:name ?drg_name. 11 } </pre>	
<div> <div>Run to File</div> <div>Text</div> <div>Charts</div> <div>Visualize</div> <div>787 Results, 380 ms</div> </div>	
drg_name	conditions
"Pramlintide"	"type 1 diabetes mellitus treatment adjunct, maturity-onset diabetes of the young type 2, type-1 diab.
"Halcinonide"	"discoid lupus erythematosus, hand dermatosis, inflammation, scalp dermatosis, leg dermatosis, facia.
"Furosemide"	"renal disease with edema, congestive heart failure, chronic kidney disease, nephrotic syndrome, liver.
"Ergotamine"	"cluster headache prevention, migraine disorder"
"Cyanocobalamin"	"prevention of vitamin b12 deficiency, multiple sclerosis, alcoholic neuropathy, pernicious anemia, dia.
"Nelarabine"	"t-cell lymphoblastic lymphoma, precursor T-cell acute lymphoblastic leukemia, precursor T-lymphob.
"Tizanidine"	"muscle spasticity of spinal origin, trigeminal neuralgia, lower back pain, spasm, pain, inflammatory m.
"Nitrofurantoin"	"staphylococcus saprophyticus urinary tract infection, urinary tract infection, gram-negative bacterial .
"Abatacept"	"rheumatoid arthritis, polyarthritis, arthritis"
"Obeticholic acid"	"primary biliary cholangitis, cholangitis, liver cirrhosis, primary biliary cirrhosis"
"Eplerenone"	"chronic heart failure following myocardial infarction, arterial hypertension"
"Amprenavir"	"HIV/AIDS, human immunodeficiency virus infectious disease"
"Corticotropin"	"optic neuritis, systemic dermatomyositis, stevens-johnson syndrome, multiple sclerosis, infantile epil.
"Methazolamide"	"secondary glaucoma, glaucoma, open-angle glaucoma, angle-closure glaucoma"
"Lodoxamide"	"giant papillary conjunctivitis, keratitis, conjunctivitis, keratoconjunctivitis, vernal conjunctivitis"
"Mafenide"	"infection of burn wound, bacterial infectious disease"

3. Find the drug that treats max and unique conditions

SELECT ?drg_name (COUNT(DISTINCT ?cnd_name) AS ?Count_conditions)

WHERE{

 ?treat a tts:Treatment;

 tts:id ?treat_id;

 tts:ref_condition_id ?ref_cnd_id;

 tts:ref_drug_id ?ref_drg_id.

 ?ref_cnd_id a cnd:Condition;

 cnd:name ?cnd_name.

 ?ref_drg_id a drg:Drug;


 drg:name ?drg_name.

```
}
```

```
GROUP BY ?drg_name
```

```
ORDER BY DESC (?Count_conditions)
```

```
LIMIT 1
```



The screenshot shows a query editor interface with a dark theme. The top bar includes buttons for 'Run', 'Show Plan', 'Reasoning', and a search icon. The main area displays a SQL query with line numbers 1 through 14. The query is a SELECT statement with a WHERE clause containing several joins and a GROUP BY clause. The bottom bar shows 'Run to File', 'Text', 'Charts', and 'Visualize' buttons, along with '1 Results, 360 ms'. Below this, a table displays the results of the query.

```
1 SELECT ?drg_name (COUNT(DISTINCT ?cnd_name) AS ?Count_conditions)
2 WHERE{
3     ?treat a tts:Treatment;
4         tts:id ?treat_id;
5         tts:ref_condition_id ?ref_cnd_id;
6         tts:ref_drug_id ?ref_drg_id.
7     ?ref_cnd_id a cnd:Condition;
8         cnd:name ?cnd_name.
9     ?ref_drg_id a drg:Drug;
10        drg:name ?drg_name.
11 }
12 GROUP BY ?drg_name
13 ORDER BY DESC (?Count_conditions)
14 LIMIT 1
```

drg_name	Count_conditions
"Methotrexate"	43

```
SELECT ?drg_name (COUNT(DISTINCT ?cnd_name) AS ?Count_conditions)
```

```
WHERE{
```

```
    ?treat a tts:Treatment;
```

```
        tts:id ?treat_id;
```

```
        tts:ref_condition_id ?ref_cnd_id;
```

```
        tts:ref_drug_id ?ref_drg_id.
```

```
    ?ref_cnd_id a cnd:Condition;
```

```
        cnd:name ?cnd_name.
```

```
    ?ref_drg_id a drg:Drug;
```

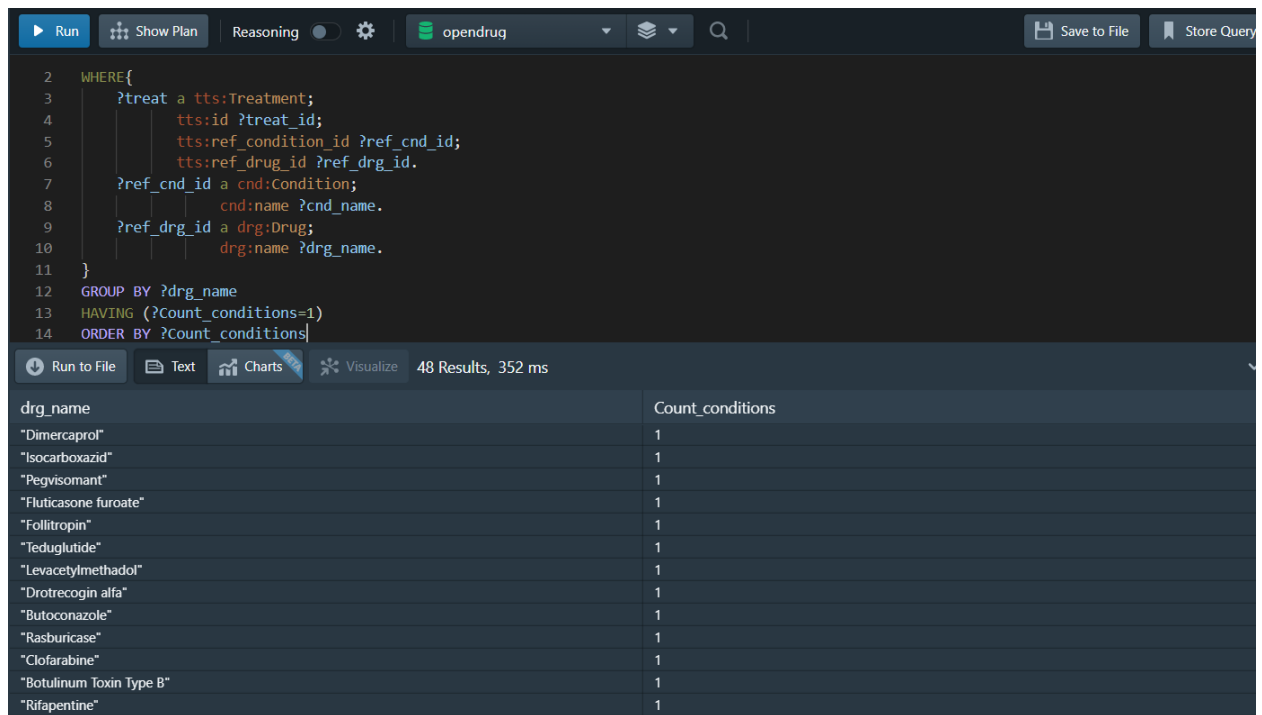
```
        drg:name ?drg_name.
```

}

GROUP BY ?drg_name

HAVING (?Count_conditions=1)

ORDER BY ?Count_conditions



The screenshot shows a query editor interface with a dark theme. The top bar includes buttons for 'Run', 'Show Plan', 'Reasoning', and a settings icon, along with a dropdown menu set to 'opendrug'. The query text is as follows:

```
2 WHERE{
3   ?treat a tts:Treatment;
4       tts:id ?treat_id;
5       tts:ref_condition_id ?ref_cnd_id;
6       tts:ref_drug_id ?ref_drg_id.
7   ?ref_cnd_id a cnd:Condition;
8       cnd:name ?cnd_name.
9   ?ref_drg_id a drg:Drug;
10      drg:name ?drg_name.
11 }
12 GROUP BY ?drg_name
13 HAVING (?Count_conditions=1)
14 ORDER BY ?Count_conditions
```

Below the query, the interface shows '48 Results, 352 ms'. The results are displayed in a table with two columns: 'drg_name' and 'Count_conditions'.

drg_name	Count_conditions
"Dimercaprol"	1
"Isocarboxazid"	1
"Pegvisomant"	1
"Fluticasone furoate"	1
"Follitropin"	1
"Ieduglutide"	1
"Levacetylmethadol"	1
"Drotrecogin alfa"	1
"Butoconazole"	1
"Rasburicase"	1
"Clofarabine"	1
"Botulinum Toxin Type B"	1
"Rifapentine"	1

4. Expensive drugs and the various conditions they treat

SELECT ?pdt_name ?drg_name ?type ?store_id (?price as ?max_price)

WHERE{

?p a price:Price;

price:id ?id;

price:price ?price;

```

price:store_id ?store_id;

price:type ?type;

price:ref_product_id ?ref_pdt_id.

?ref_pdt_id a pdt:Product;

    pdt:name ?pdt_name;

    pdt:ref_drug_id ?ref_drg_id.

?ref_drg_id a drg:Drug;

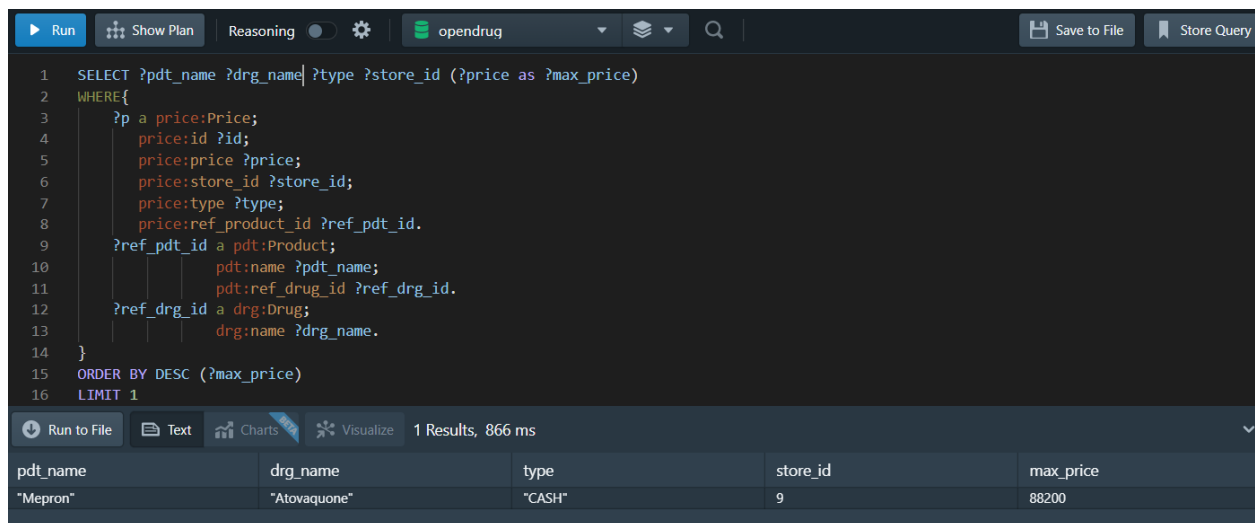
    drg:name ?drg_name.

}

ORDER BY DESC (?max_price)

LIMIT 1

```



The screenshot shows a SPARQL query editor interface. The query is as follows:

```

1 SELECT ?pdt_name ?drg_name | ?type ?store_id (?price as ?max_price)
2 WHERE{
3     ?p a price:Price;
4     price:id ?id;
5     price:price ?price;
6     price:store_id ?store_id;
7     price:type ?type;
8     price:ref_product_id ?ref_pdt_id.
9     ?ref_pdt_id a pdt:Product;
10         pdt:name ?pdt_name;
11         pdt:ref_drug_id ?ref_drg_id.
12     ?ref_drg_id a drg:Drug;
13         drg:name ?drg_name.
14 }
15 ORDER BY DESC (?max_price)
16 LIMIT 1

```

The results pane shows 1 result in 866 ms:

pdt_name	drg_name	type	store_id	max_price
"Mepron"	"Atovaquone"	"CASH"	9	88200

5. The most popular type of payment?

```
SELECT ?type (COUNT(?type) as ?count_type)
```

```
WHERE{
```

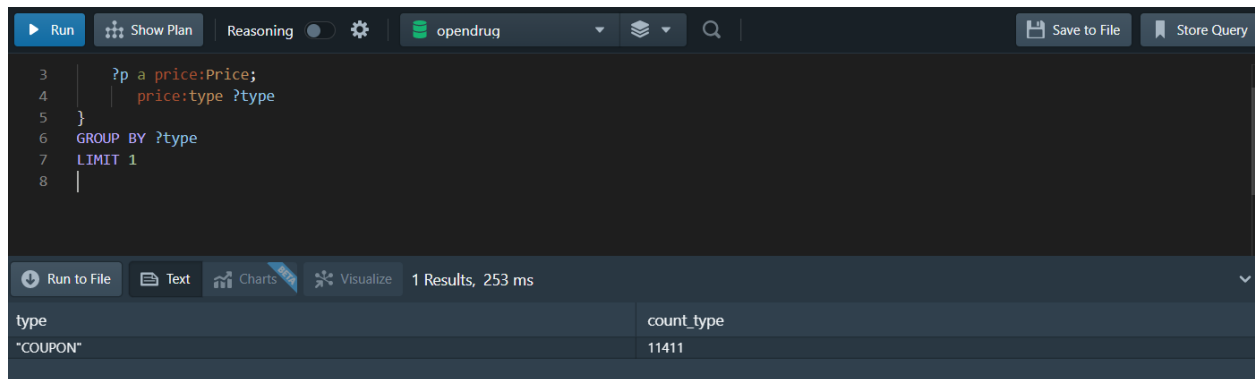
```
    ?p a price:Price;
```

```
    price:type ?type
```

```
}
```

```
GROUP BY ?type
```

```
LIMIT 1
```



The screenshot shows a SPARQL query editor interface. The query is as follows:

```
3 | ?p a price:Price;
4 |   price:type ?type
5 | }
6 | GROUP BY ?type
7 | LIMIT 1
8 |
```

The interface includes a toolbar with buttons for 'Run', 'Show Plan', 'Reasoning', and 'Settings'. The 'Run' button is highlighted. Below the query editor, there is a status bar indicating '1 Results, 253 ms'. The results are displayed in a table with two columns: 'type' and 'count_type'.

type	count_type
"COUPON"	11411

6. Find stores that accept gold as a payment

```
SELECT ?str_name ?type
```

```
WHERE{
```

```
    ?p a price:Price;
```

```
    price:ref_store_id ?ref_str_id;
```

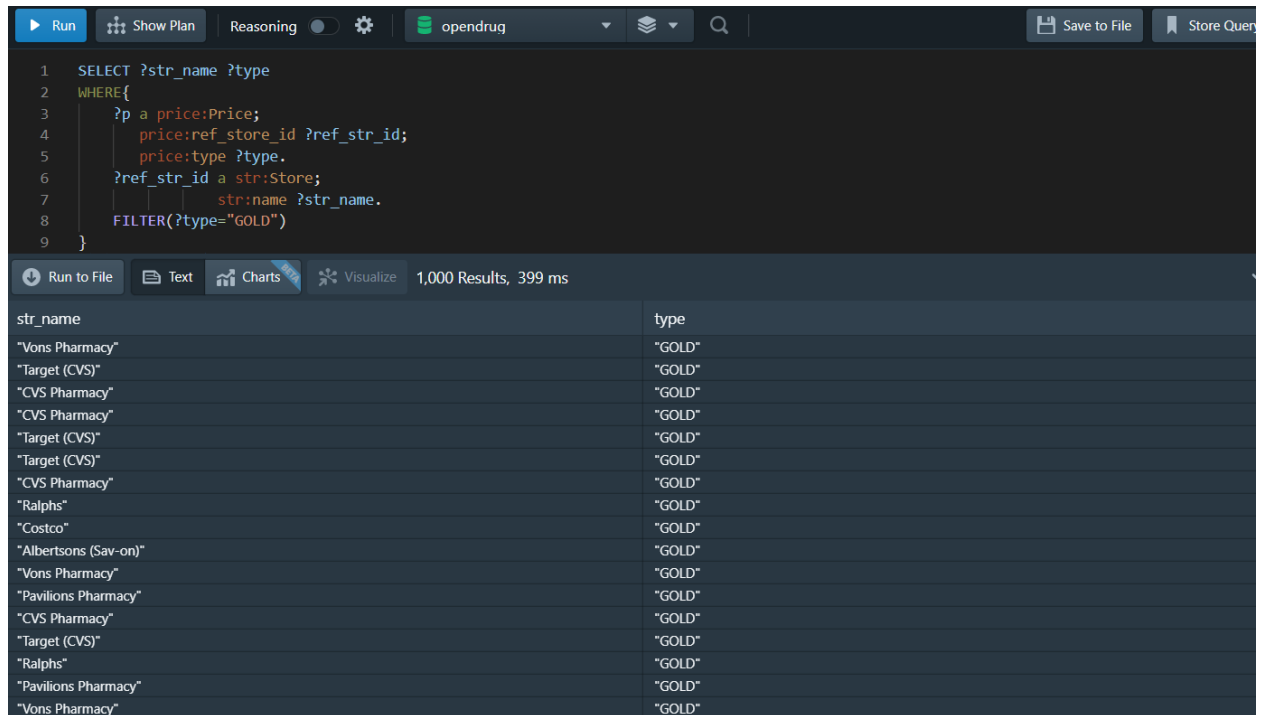
```
    price:type ?type.
```

```
    ?ref_str_id a str:Store;
```


str:name ?str_name.

FILTER(?type="GOLD")

}



The screenshot shows a SPARQL query editor interface. The query is as follows:

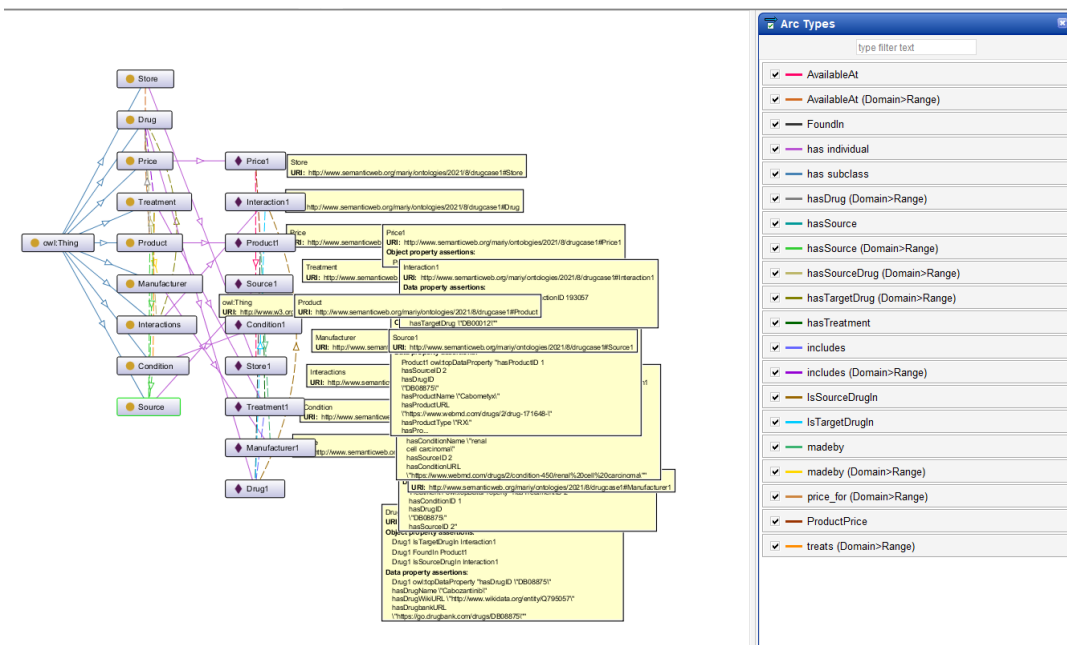
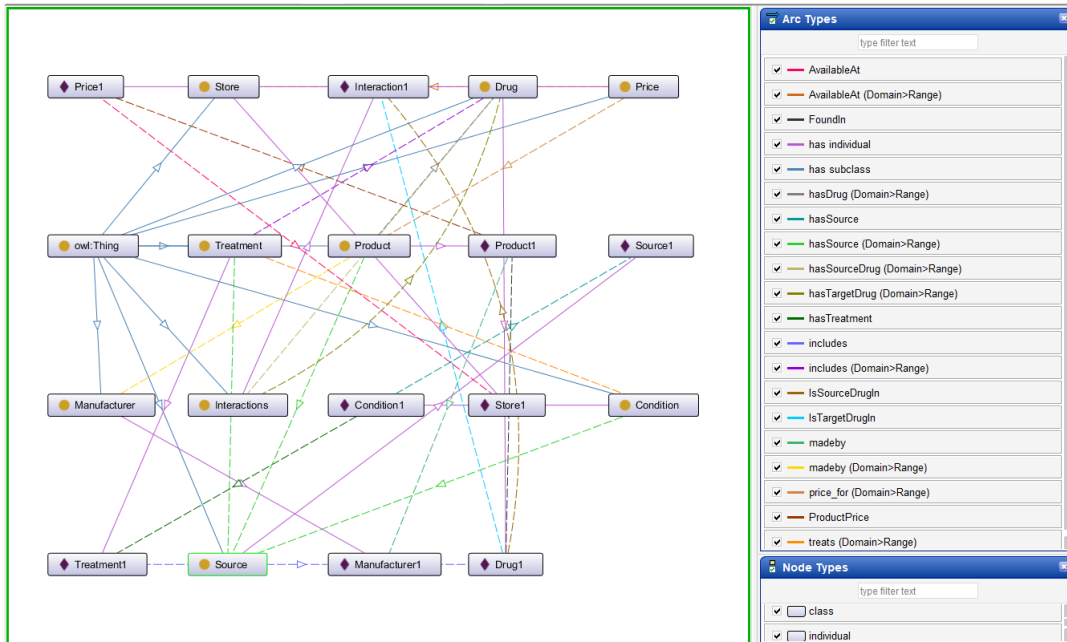
```
1 SELECT ?str_name ?type
2 WHERE{
3   ?p a price:Price;
4     price:ref_store_id ?ref_str_id;
5     price:type ?type.
6   ?ref_str_id a str:Store;
7     str:name ?str_name.
8   FILTER(?type="GOLD")
9 }
```

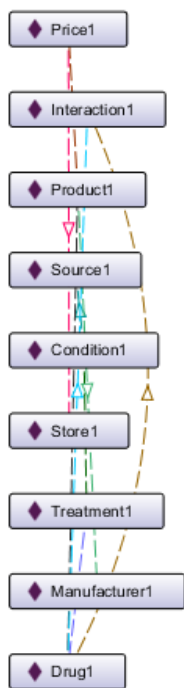
The results section shows 1,000 results in 399 ms. The table below represents the first 15 rows of the results:

str_name	type
"Vons Pharmacy"	"GOLD"
"Target (CVS)"	"GOLD"
"CVS Pharmacy"	"GOLD"
"CVS Pharmacy"	"GOLD"
"Target (CVS)"	"GOLD"
"Target (CVS)"	"GOLD"
"CVS Pharmacy"	"GOLD"
"Ralphs"	"GOLD"
"Costco"	"GOLD"
"Albertsons (Sav-on)"	"GOLD"
"Vons Pharmacy"	"GOLD"
"Pavilions Pharmacy"	"GOLD"
"CVS Pharmacy"	"GOLD"
"Target (CVS)"	"GOLD"
"Ralphs"	"GOLD"
"Pavilions Pharmacy"	"GOLD"
"Vons Pharmacy"	"GOLD"

Protege

A basic blueprint for the dataset was constructed with a few example instances to check if the flow functions. The following screenshots are a detailed view of the ontology.





P	Price1		URI: http://www.semanticweb.org/mariy/ontologies/2021/8/drugcase1#Price1
	Object property assertions:		Price1 AvailableAt Store1
	Interaction1		URI: http://www.semanticweb.org/mariy/ontologies/2021/8/drugcase1#Interaction1
	Data property assertions:		Interaction1 owl:topDataProperty "has InteractionID 193057"
S	Source1		URI: http://www.semanticweb.org/mariy/ontologies/2021/8/drugcase1#Source1
	Product1 owl:topDataProperty "has ProductID 1"		hasSourceID 2
	hasDrugID		"DB08875"
	hasProductURL		"https://www.webmd.com/drugs/2/drug-171648-1"
C	hasConditionName		"renal cell carcinoma"
	hasSourceID 2		hasConditionURL
	hasConditionURL		"https://www.webmd.com/drugs/2/condition-450/renal%20cell%20carcinoma"
	URI: http://www.semanticweb.org/mariy/ontologies/2021/8/drugcase1#Manufacturer1		hasDrugID 1
D	hasDrugID		"DB08875"
	hasSourceID 2"		Object property assertions:
	Drug1 is TargetDrugIn Interaction1		Drug1 FoundIn Product1
	Drug1 is SourceDrugIn Interaction1		Data property assertions:
Drug1 owl:topDataProperty "hasDrugID "DB08875"		hasDrugName	"Cabozantinib"
hasDrugWikiURL		"http://www.wikidata.org/entity/Q795057"	hasDrugbankURL
"		"https://go.drugbank.com/drugs/DB08875"	

Result/inference

The training took us about 6 weeks and we learnt tools like Apache Nifi, Stardog, Protege and the query language SPARQL. After a rigorous couple of weeks, we had the opportunity to integrate our knowledge across all the tools into one project where we chose a dataset and injected it from Apache nifi by converting CSV files to JSON and into Stardog using SMS files and queried the data using SPARQL. We also created a basic ontology of the data set on Protege.

Some of the insights generated from the data set are:

1. About 1000 stores accept gold as a payment method and gold is the second most preferred method of payment for drugs.
2. Coupons are the most preferred method of payment, however, there is no exact evidence as to why.
3. The Most Expensive Product is Mepron containing Atovaquone was once sold for 88200 paid all in cash (We are not sure if the data is inconsistent here but there is no specification about the currency)
4. We did a small survey on which drugs were widely used to treat many conditions and which cured only one. The most common drug is Methotrexate used to treat 43 different conditions whereas there are about 48 drugs that treat one unique condition.
5. Drugs are used to treat multiple conditions and can be found as part of our query 2.
6. This leads to our final point that the same drug is used in multiple products aimed to treat different conditions.

References

1. <https://towardsdatascience.com/knowledge-graph-bb78055a7884>
2. <https://warwick.ac.uk/fac/soc/ces/research/current/socialtheory/maps/ology/>
3. <https://protege.stanford.edu/>
4. <https://www.stardog.com/platform/features/connect-data-silos/>
5. Dataset: <https://www.kaggle.com/mannbrinson/open-drug-knowledge-graph>
6. <https://nifi.apache.org/docs/nifi-docs/html/overview.html>