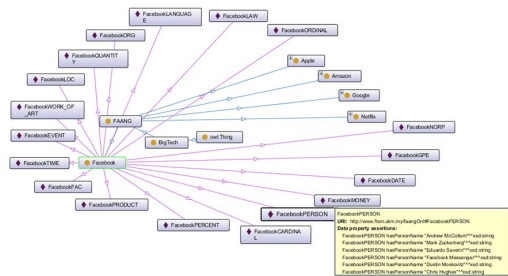


# **Named Entity Extraction in RDF/RDFS for Big Tech**

**Lim Teck Huat (P107443)**

## Topic

Review Named Entity Extraction (NEE) and how NEE can form the taxonomy. Apply one approach of NEE based on one domain and translate it into turtle format of RDF/RDFS data model.

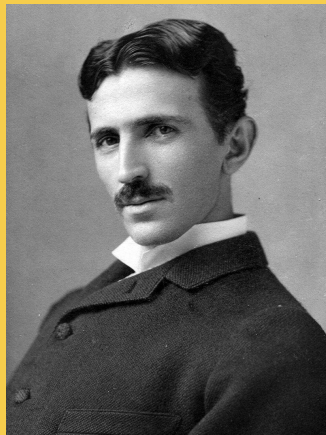


# Abstract

The study aims to propose a pipeline that performs Named Entity Extraction using spaCy on raw unstructured Big Tech Wikipedia dataset and convert it to RDF/RDFS format. The simple pipeline was without fine-tuning on the Wikipedia dataset, only fine-tuning done was on the size of the pre-trained model where it ranges from small, medium and big. Competency questions for verification were conducted pre-extraction and SPARQL competency questions were conducted post-extraction for validation. The extracted entities were converted into RDF/RDFS format containing prefixes, classes, properties and individuals where the extracted entities mainly form the RDF part of the Big Tech taxonomy. It was found that the out-off-the-box medium sized spaCy model with Precision, Recall and F-score of 0.85, 0.84 and 0.85 performs satisfactorily on the Big Tech dataset with an average global Precision, Recall and F-score of 0.83, 0.80 and 0.81 respectively with respect to algorithm aided self-developed gold standard.

# 1. Introduction

Prior to investing in something, it is a good idea to do some **background research**, more so if it involves a huge amount risks and money such as in stocks specifically technological company stocks. **Nikola is a billion dollar zero emission truck public tech company** founded by Trevor Milton and was supposed to rival Tesla. Hindenburg Research (2020) highlighted that using concept truck alone it was worth **20 billion dollars** and was built from **dozens of lies**. **Recommended key words or sentences** in the form of entities can be obtained by applying pre-trained Named Entity Extraction on **publicly available datasets** for company of interest to **aid background research**.



# 1. Introduction (cont'd)

Named Entity Extraction can be used to extract the **hypernym** which is a **relatively general term** such as Facebook and hyponym which is a **relatively specific term** such as FacebookPERSON, Facebook GPE and others. **Relatively**, going down the hierarchy, FacebookPERSON is the hypernym while entities extracted such as “Mark Zuckerberg” is the hyponym. Using NEE, **a taxonomy or concept hierarchy can be formed** as shown in figure on the left.

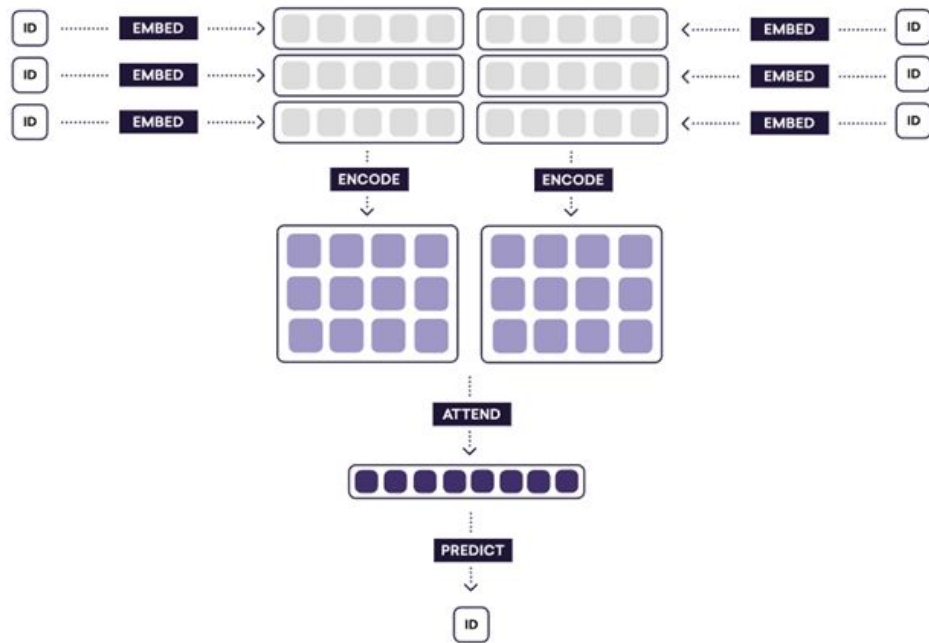


In this study, NEE was used for **taxonomy extraction** mainly for **co-hyponyms** or **Abox** for individual tech companies with highest frequency hypernym extracted from the pipeline. This study employed a **modular approach** whereby additional input data can be plugged in to further populate the taxonomy.

# 1. Introduction (cont'd)

The Stanford Natural Language Processing Group (2022) explained that Named Entity Recognition (NER) is used to detect and classify sequences of words in unstructured text into classes of mainly person, organization and location.

In this study, spaCy was employed for NER followed by extraction, the library is neural network based, it is an implementation of embed, encode, attend and predict. Embed is a process where binarization from sparse data into dense vectors happens and then using encode, the data is converted from vector to matrix using CNN in spaCy's case, this is continued by attend where it performs contextual vectorization using attention matrix to compress the data and finally predict where all these data are fed into a multi-layer perceptron neural network for prediction as shown in figure on the left.

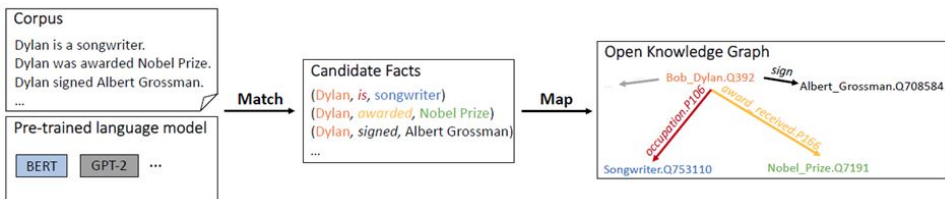


# 1. Introduction (cont'd)

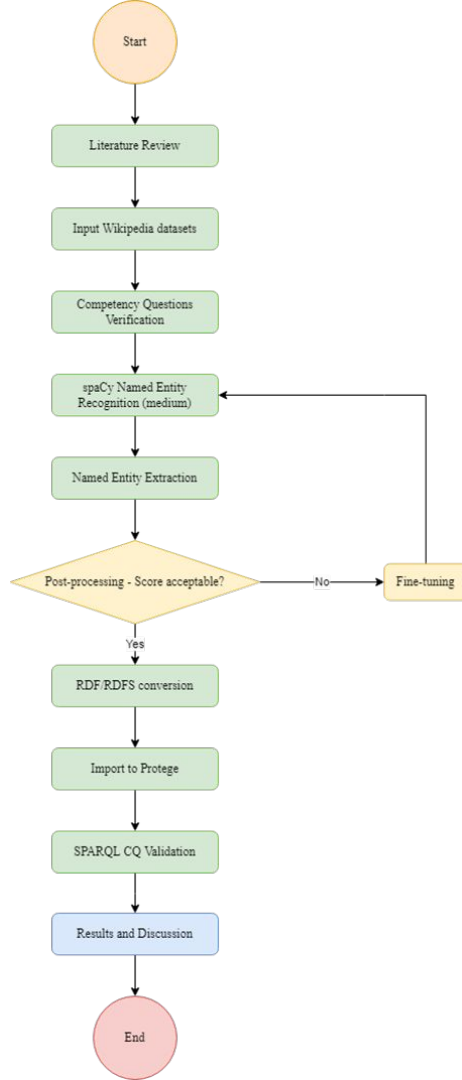
Shelar et al. (2020) experimented using advance search panel dataset of IBM with multiple libraries for Named Entity Recognition and concluded that spaCy obtains the best score and speed versus the rest i.e Apache OpenNLP and TensorFlow while Chantrapornchai & Tunsakul (2021) used crawled tourism (TripAdvisor, Traveloka and Hotels.com) domain data demonstrated that BERT works slightly better with clean data relative to spaCy.

This study follows the concept of zero-shot learning by Wang et al. (2020) where no fine-tuning was used on the off-the shelf spaCy model since BERT requires extensive cleaning, the concept of zero-shot learning is illustrated in figure on the left.

Mikheev et al. (1999) demonstrated a rule and statistical based hybrid model for Named Entity Recognition, the concept of Partial Match and self-added Exact Match was —instead applied as an evaluation technique.







## 2. Materials and Method

### Dataset

- Wikipedia dataset of FAANG.

### Flow of the pipeline

- Literature Review.
- Input Wikipedia datasets.
- CQ Verification.
- NER.
- NEE.
- Scoring.
- RDF/RDFS translation.
- Protege import.
- SPARQL CQ Validation.
- Results and Discussion.
- End.

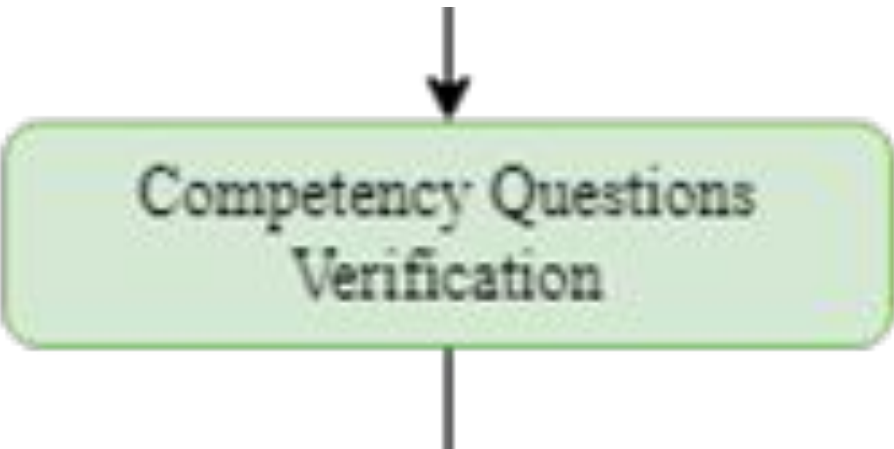


## 2.1 Input Wikipedia datasets

Figure below shows the code for generating .owl type files to facilitate Protégé import, updating the code to .txt works as well.

### ▼ Open and Load

```
[ ] 1 #open the file
    2 faang_1 = "/content/drive/MyDrive/Colab Notebooks/OKR_NEE_datasets/1_Facebook.txt"
    3 file1 = open(faang_1, "r")
    4 #read the file
    5 Facebook = file1.read()
    6 Facebook
```



## Competency Questions Verification

### 2.2 CO Verification

Competency questions of 5 numbers were asked to probe the extent of the data.

Q1. Who are the founder(s) of respective FAANG?

Q2. Where are they founded?

Q3. When are they founded?

Q4. What are their product(s)?

Q5. What is the nationality of the company?

## 2.3 spaCy NER

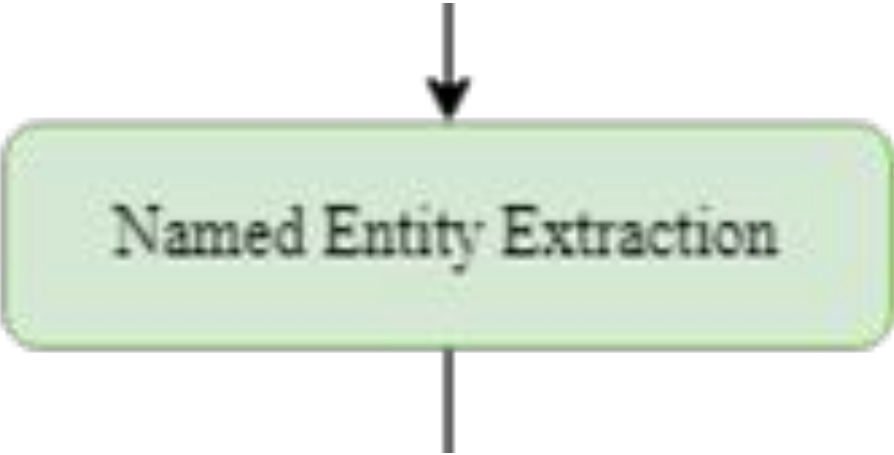
```
graph TD; A[ ] --> B[spaCy Named Entity Recognition (medium)]; B --> C[ ]
```

spaCy Named Entity Recognition (medium)

```
[ ] 1 #load spacy
    2 NER_1 = spacy.load("en_core_web_md")
```

spaCy and displaCy

```
[ ] 1 #get the labels and entities
    2 Facebookdoc = NER_1(Facebook)
    3
```



## Named Entity Extraction

## 2.4 spaCy NEE

Entities, labels, start and end positions of the entities can all be extracted using the code as shown in figure below along with highest frequency entities. The extracted entities mainly populates the RDF part of the Big Tech domain taxonomy.

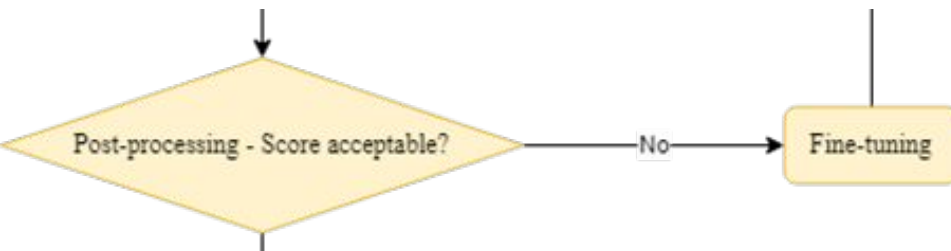
spaCy and displaCy

```
[ ] 1 #get the labels and entities
2 Facebookdoc = NER_1(Facebook)
3
4 Facebookentities = []
5 Facebooklabels = []
6 Facebookstartpos = []
7 Facebookendpos = []
8
9 for ent in Facebookdoc.ents:
10     Facebookentities.append(ent)
11     Facebooklabels.append(ent.label_)
12     Facebookstartpos.append(ent.start_char)
13     Facebookendpos.append(ent.end_char)
14
15 Facebookdf = pd.DataFrame({'Entities':Facebookentities,'Labels':Facebooklabels, 'Position_Start':Facebookstartpos, 'Position_End':Facebookendpos})
16
17 Facebookdf
18
19 Facebookdf.to_csv(r'./content/drive/MyDrive/Colab Notebooks/OKR_NEE_datasets/out.csv', index = False)

[ ] 1 #Count
2 items_1 = [x.text for x in Facebookdoc.ents]
3 y_1 = Counter(items_1).most_common(5) #top 5

[ ] 1 y_1[0][0] #highest frequency
'Facebook'

[ ] 1 #visualization
2 doc_1 = NER_1(Facebook)
3 displaCy.render(doc_1, style="ent", jupyter=True)
```



## 2.5 Score

With respect to self-developed gold standard, global and local scores consisting of Precision, Recall and F-score were extracted. Figure below shows the codes necessary for scoring.

```
▼ Score

[ ] 1 #equate
    2 nlp_1 = NER_1

[ ] 1 #working code
    2
    3 Facebookexamples = []
    4
    5 data_1 = [(Facebook, {"entities": [(0,8,'ORG'), (15,23,'NORP'), (83,97,'ORG'), (109,113,'DATE'), (117,132,'PERSON'),
    6 (145,160,'ORG'), (184,199,'PERSON'), (201,216,'PERSON'), (218,234,'PERSON'), (240,252,'PERSON'),
    7 (315,323,'NORP'), (381,388,'ORG'), (428,442,'NORP'), (467,471,'DATE'), (480,497,'DATE'),
    8 (505,509,'DATE'), (511,519,'ORG'), (528,539,'CARDINAL'), (540,547,'DATE'), (573,580,'ORDINAL'),
    9 (648,657,'DATE'), (659,667,'ORG'), (1085,1103,'PRODUCT'), (1187,1195,'ORG'), (1266,1274,'ORG'),
   10 (1347,1366,'ORG'), (1418,1422,'DATE'), (1423,1427,'GPE'), (1639,1647,'ORG')])])
   11
   12 for text_1, annots_1 in data_1:
   13     doc_1 = nlp_1.make_doc(text_1)
   14     Facebookexamples.append(Example.from_dict(doc_1, annots_1))
   15
   16 #output to txt
   17 f = sys.stdout
   18 f = open('score_output.txt', 'a') #append to opened common file
   19
   20 print("#Facebook score .", file=f)
   21 print(nlp_1.evaluate(Facebookexamples), file=f) #Global and local entity metrics
   22 f.close() #close file

[ ] 1 print(nlp_1.evaluate(Facebookexamples))

{'token_acc': 1.0, 'token_p': 1.0, 'token_r': 1.0, 'token_f': 1.0, 'tag_acc': None, 'sents_p': None, 'sents_r': None, 'se
```



## 2.6 RDF/RDFS conversion

Basically, the general idea is to transform for individual 18 numbers of spaCy labels and extracted entities to be RDF/RDFS compatible.

```
1 #For Facebook
2 |
3 #output to owl
4 f = sys.stdout
5 f = open('p107443_output.owl', 'a') #append to opened common file
6
7 PropLabel_1 = y_1[0][0]
8
9 #General Prefixes
10 print("#Prefixes", file=f)
11 print("@prefix faang: <http://www.ftsm.ukm.my/faangOnt#> .", file=f)
12 print("@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .", file=f)
13 print("@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .", file=f)
14 print("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .", file=f)
15
16 #General Class and SubClass
17 print("\n#Classes and SubClasses", file=f)
18 print("faang:BigTech rdf:type rdfs:Class .", file=f) #Class
19 print("faang:FAANG rdf:type rdfs:Class .", file=f) #Class
20 print("faang:FAANG rdfs:subClassOf faang:BigTech .", file=f) #SubClass
21 print("faang:{proplab_1} rdf:type rdfs:Class .".format(proplab_1=PropLabel_1), file=f) #Class
22 print("faang:{proplab_1} rdfs:subClassOf faang:FAANG .".format(proplab_1=PropLabel_1), file=f) #Facebook SubClass
23
24 #Facebook Individuals
25 print("\n#Facebook Individuals", file=f)
26 print("faang:{proplab_1}PERSON rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
27 print("faang:{proplab_1}NORP rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
28 print("faang:{proplab_1}FAC rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
29 print("faang:{proplab_1}ORG rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
30 print("faang:{proplab_1}GPE rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
31 print("faang:{proplab_1}LOC rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
32 print("faang:{proplab_1}PRODUCT rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
```



## 2.6 RDF/RDFS conversion (cont'd)

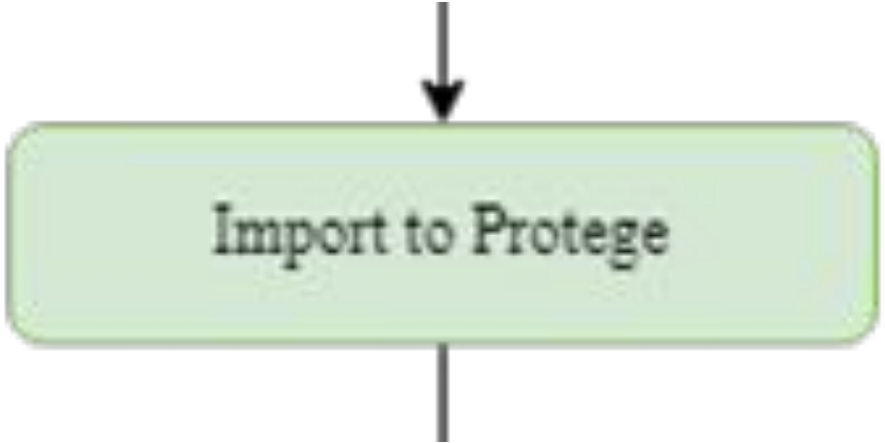
```
45 #General spaCy labels to Data Properties
46 print("\n#spaCy labels", file=f)
47 PERSON = "hasPersonName" #1
48 NORP = "hasNationalityReligiousPolitical" #2
49 FAC = "hasBuildingsAirportsHighwaysBridges" #3
50 ORG = "hasCompaniesAgenciesInstitutions" #4
51 GPE = "hasCountriesCitiesState" #5
52 LOC = "hasLocation" #6
53 PRODUCT = "hasProduct" #7
54 EVENT = "hasEvent" #8
55 WORK_OF_ART = "hasWorkOfArt" #9
56 LAW = "hasLaw" #10
57 LANGUAGE = "hasLanguage" #11
58 DATE = "hasDate" #12
59 TIME = "hasTime" #13
60 PERCENT = "hasPercent" #14
61 MONEY = "hasMoney" #15
62 QUANTITY = "hasQuantity" #16
63 ORDINAL = "hasOrdinal" #17
64 CARDINAL = "hasCardinal" #18
65
```

```
66 #General Data Properties
```

```
67 print("\n#Data Properties", file=f)
68 print("faang:{} rdf:type rdf:Property ." .format(PERSON), file=f) #1
69 print("faang:{} rdfs:range xsd:string ." .format(PERSON), file=f) #1a
70 print("faang:{} rdf:type rdf:Property ." .format(NORP), file=f) #2
71 print("faang:{} rdfs:range xsd:string ." .format(NORP), file=f) #2a
72 print("faang:{} rdf:type rdf:Property ." .format(FAC), file=f) #3
73 print("faang:{} rdfs:range xsd:string ." .format(FAC), file=f) #3a
74 print("faang:{} rdf:type rdf:Property ." .format(ORG), file=f) #4
```

```
105 #Specific Facebook Triples
106 print("\n#Facebook Triples", file=f)
107 for i in range(len(FacebooklistEntities)):
108     if FacebooklistLabels[i]=='PERSON':
109         print('faang:{proplab_1}PERSON faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasPersonName', ent=FacebooklistEntities[i]), file=f)
110     elif FacebooklistLabels[i]=='NORP':
111         print('faang:{proplab_1}NORP faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasNationalityReligiousPolitical', ent=FacebooklistEntities[i]), file=f)
112     elif FacebooklistLabels[i]=='FAC':
113         print('faang:{proplab_1}FAC faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasBuildingsAirportsHighwaysBridges', ent=FacebooklistEntities[i]), file=f)
114     elif FacebooklistLabels[i]=='ORG':
115         print('faang:{proplab_1}ORG faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasCompaniesAgenciesInstitutions', ent=FacebooklistEntities[i]), file=f)
116     elif FacebooklistLabels[i]=='GPE':
117         print('faang:{proplab_1}GPE faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasCountriesCitiesState', ent=FacebooklistEntities[i]), file=f)
118     elif FacebooklistLabels[i]=='LOC':
119         print('faang:{proplab_1}LOC faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasLocation', ent=FacebooklistEntities[i]), file=f)
120     elif FacebooklistLabels[i]=='PRODUCT':
121         print('faang:{proplab_1}PRODUCT faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasProduct', ent=FacebooklistEntities[i]), file=f)
122     elif FacebooklistLabels[i]=='EVENT':
123         print('faang:{proplab_1}EVENT faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasEvent', ent=FacebooklistEntities[i]), file=f)
124     elif FacebooklistLabels[i]=='WORK_OF_ART':
125         print('faang:{proplab_1}WORK_OF_ART faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasWorkOfArt', ent=FacebooklistEntities[i]), file=f)
126     elif FacebooklistLabels[i]=='LAW':
127         print('faang:{proplab_1}LAW faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasLaw', ent=FacebooklistEntities[i]), file=f)
128     elif FacebooklistLabels[i]=='LANGUAGE':
129         print('faang:{proplab_1}LANGUAGE faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasLanguage', ent=FacebooklistEntities[i]), file=f)
130     elif FacebooklistLabels[i]=='DATE':
131         print('faang:{proplab_1}DATE faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasDate', ent=FacebooklistEntities[i]), file=f)
132     elif FacebooklistLabels[i]=='TIME':
133         print('faang:{proplab_1}TIME faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasTime', ent=FacebooklistEntities[i]), file=f)
```



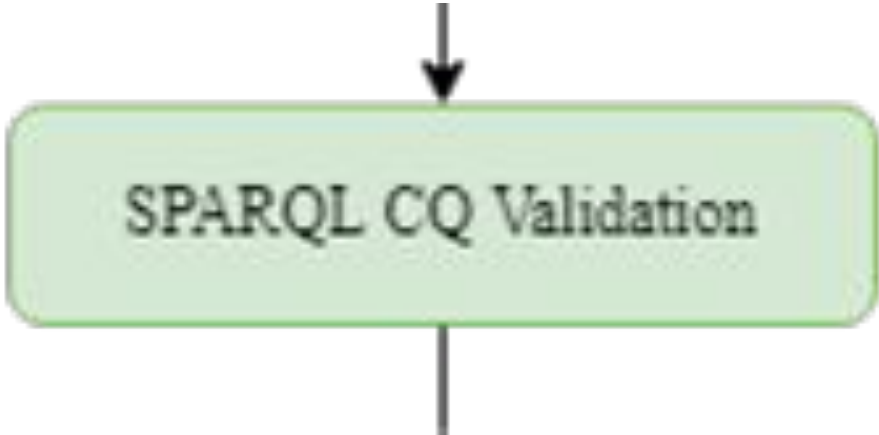


Import to Protege

## 2.7 Import to Protégé

Figure below shows the code for generating .owl type files to facilitate Protégé import, updating the code to .txt works as well.

```
1 #For Facebook
2
3 #output to owl
4 f = sys.stdout
5 f = open('p107443_output.owl', 'a') #append to opened common file
```



## SPARQL CQ Validation

## 2.8 SPARQL queries CQ Validation

```
#Fixed
PREFIX faang: <http://www.ftsm.ukm.my/faangOnt#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?subject ?object
Q1. Who are the founder(s) of respective FAANG?
#Variable
WHERE
{
  ?subject rdf:type faang:Facebook . #swap "Facebook" for subsequent {F}AANGs
  ?subject faang:hasPersonName ?object .
}
Q2. Where are they founded?
#Variable
WHERE
{
  ?subject rdf:type faang:Facebook . #swap "Facebook" for subsequent {F}AANGs
  ?subject faang:hasCountriesCitiesState ?object .
}
Q3. When are they founded?
#Variable
WHERE
{
  ?subject rdf:type faang:Facebook . #swap "Facebook" for subsequent {F}AANGs
  ?subject faang:hasDate ?object .
}
Q4. What are their product(s)?
#Variable
WHERE
{
  ?subject rdf:type faang:Facebook . #swap "Facebook" for subsequent {F}AANGs
  ?subject faang:hasCompaniesAgenciesInstitutions ?object . #for all FAANGs
  #include hasPersonName for Amazon
  #include hasProduct for Apple
  #include hasPersonName for Google
  #include hasBuildingsAirportsHighwaysBridges for Google
}
Q5. What is the nationality of the company?
#Variable
WHERE
{
  ?subject faang:hasNationalityReligiousPolitical ?object .
}
```

**Table 2.** Out-of-the-box Precision, Recall and F-score

Size	ENTS_P	ENTS_R	ENTS_F
Small	0.85	0.84	0.84
Medium	0.85	0.84	0.85
Large	0.86	0.85	0.85

Reference: [spaCy, 2022](#)

**Table 3.** Overall Precision, Recall and F-score

FAANG	Precision	Recall	F-score
Facebook	0.846154	0.758621	0.800000
Amazon	0.836735	0.803922	0.820000
Apple	0.875000	0.851351	0.863014
Netflix	0.882353	0.923077	0.902256
<b>Google</b>	<b>0.730159</b>	<b>0.638889</b>	<b>0.681481</b>
Average	0.8340802	0.795172	0.8133502

**Table 4.** Facebook Precision, Recall and F-score

Labels	Precision	Recall	F-score
NORP	1.000000	1.000000	1.000000
<b>ORG</b>	<b>0.833333</b>	<b>0.500000</b>	<b>0.625000</b>
DATE	0.857143	0.857143	0.857143
PERSON	0.833333	1.000000	0.909091
CARDINAL	1.000000	1.000000	1.000000
ORDINAL	1.000000	1.000000	1.000000
<b>GPE</b>	<b>0.500000</b>	<b>1.000000</b>	<b>0.666667</b>
<b>PRODUCT</b>	<b>0.000000</b>	<b>0.000000</b>	<b>0.000000</b>

### 3. Results and Discussion

Table 2 shows that the model has an off-the-shelf Precision, Recall and F-score of 0.85, 0.84 and 0.85 respectively. Based on my observation, the values shown are ceiling values as average scores obtained did not exceed the values shown for Medium size models.

Table 3 shows the global scores of respective FAANG where the highest F-score peaks at 0.90 for Netflix and bottoms at 0.68 for Google. These scores are computed with respect to the gold standard.

Tables 4 shows the individual labels local scores for Facebook. It has issue with “PRODUCT”, “GPE” and “ORG” labels.

**Table 5.** Amazon Precision, Recall and F-score

Labels	Precision	Recall	F-score
ORG	0.827586	0.800000	0.813559
NORP	1.000000	1.000000	1.000000
<b>CARDINAL</b>	<b>0.333333</b>	<b>1.000000</b>	<b>0.500000</b>
GPE	1.000000	1.000000	1.000000
<b>PERSON</b>	<b>0.500000</b>	<b>1.000000</b>	<b>0.666667</b>
DATE	1.000000	1.000000	1.000000
MONEY	1.000000	1.000000	1.000000
ORDINAL	1.000000	1.000000	1.000000
<b>PRODUCT</b>	<b>0.000000</b>	<b>0.000000</b>	<b>0.000000</b>

### 3. Results and Discussion (cont'd)

Tables 5 and 6 shows the individual labels local scores for Amazon and Apple. Both of them have similar issues with “PRODUCT” and “CARDINAL” labels.

**Table 6.** Apple Precision, Recall and F-score

Labels	Precision	Recall	F-score
ORG	0.785714	0.916667	0.846154
NORP	1.000000	1.000000	1.000000
<b>MONEY</b>	<b>0.666667</b>	<b>0.666667</b>	<b>0.666667</b>
DATE	1.000000	1.000000	1.000000
ORDINAL	1.000000	1.000000	1.000000
<b>CARDINAL</b>	<b>0.500000</b>	<b>1.000000</b>	<b>0.666667</b>
PERSON	0.900000	0.692308	0.782609
<b>PRODUCT</b>	<b>1.000000</b>	<b>0.333333</b>	<b>0.500000</b>
GPE	1.000000	1.000000	1.000000

**Table 7.** Netflix Precision, Recall and F-score

Labels	Precision	Recall	F-score
ORG	0.823529	0.875000	0.848485
NORP	1.000000	1.000000	1.000000
GPE	1.000000	1.000000	1.000000
DATE	1.000000	1.000000	1.000000
PERSON	0.666667	1.000000	0.800000
CARDINAL	0.750000	0.750000	0.750000
LOC	1.000000	1.000000	1.000000
ORDINAL	1.000000	0.750000	0.857143
<b>WORK_OF_ART</b>	<b>0.000000</b>	<b>0.000000</b>	<b>0.000000</b>
MONEY	1.000000	1.000000	1.000000
<b>PRODUCT</b>	<b>0.000000</b>	<b>0.000000</b>	<b>0.000000</b>
PERCENT	1.000000	1.000000	1.000000

### 3. Results and Discussion (cont'd)

Tables 7 and 8 shows the individual labels local scores for Netflix and Google. Both of them have similar issues with “PRODUCT” label.

**Table 8.** Google Precision, Recall and F-score

Labels	Precision	Recall	F-score
ORG	0.729730	0.794118	0.760563
NORP	1.000000	1.000000	1.000000
<b>CARDINAL</b>	<b>0.000000</b>	<b>0.000000</b>	<b>0.000000</b>
DATE	1.000000	1.000000	1.000000
<b>PERSON</b>	<b>0.571429</b>	<b>0.800000</b>	<b>0.666667</b>
<b>WORK_OF_ART</b>	<b>0.000000</b>	<b>0.000000</b>	<b>0.000000</b>
<b>GPE</b>	<b>0.500000</b>	<b>1.000000</b>	<b>0.666667</b>
PERCENT	1.000000	1.000000	1.000000
<b>FAC</b>	<b>0.000000</b>	<b>0.000000</b>	<b>0.000000</b>
ORDINAL	1.000000	1.000000	1.000000
<b>PRODUCT</b>	<b>0.000000</b>	<b>0.000000</b>	<b>0.000000</b>



```
1 #visualization
2 doc_1 = MER_1(Facebook)
3 spacy.disply.render(doc_1, style='ent', jsyter=True)
```

Facebook is an **American NORP** online social media and social networking service owned by **Facebook ORG** Inc. Founded in **2004 DATE** by **Mark Zuckerberg PERSON** with fellow **Harvard College ORG** students and roommates **Eduardo Saverio PERSON**, **Andrew McCollum PERSON**, **Dustin Moskowitz PERSON**, and **Chris Hughes PERSON**, its name comes from the fact book directories often given to **American NORP** university students. Membership was initially limited to **Harvard ORG** students, gradually expanding to other **NorthAmerican NORP** universities and, since **2006 DATE**, anyone over **13 years old DATE**. As of **2020 DATE**, **Facebook ORG** claimed **2.6 billion CARDINAL** monthly **DATE** active users, and ranked **seventh ORGINAL** in global internet usage. It was the most downloaded mobile app of the **2010s DATE**. **Facebook** can be accessed from devices with Internet connectivity, such as personal computers, tablets and smartphones. After registering, users can create a profile revealing information about themselves. They can post text, photos and multimedia which are shared with any other users who have agreed to be their "friend" or, with different privacy settings, publicly. Users can also communicate directly with each other via **Facebook Messenger PERSON**, join common-interest groups, and receive notifications on the activities of their **Facebook ORG** friends and pages they follow. The subject of numerous controversies, **Facebook ORG** has often been criticized over issues such as user privacy (as with the **Cambridge Analytica GPE** data scandal), political manipulation (as with the **2016 DATE U.S. GPE** elections), mass surveillance, psychological effects such as addiction and low self-esteem, and content such as fake news, conspiracy theories, copyright infringement, and hate speech. Commentators have accused Facebook of willingly facilitating the spread of such content, as well as exaggerating its number of users to appeal to advertisers.

## 3. Results and Discussion (cont'd)

displaCy visualization.

```
1 #visualization
2 doc_2 = MER_2(Amazon)
3 spacy.disply.render(doc_2, style='ent', jsyter=True)
```

**Amazon.com, Inc. ORG** (/əˈmæzoʊn ˈAM-ˌzoʊn) is an **American NORP** multinational technology company which focuses on e-commerce, cloud computing, digital streaming, and artificial intelligence. It is one of the **Big Five CARDINAL** companies in the **U.S. GPE** information technology industry, along with **Google ORG**, **Apple ORG**, **Microsoft ORG**, and **Facebook ORG**. The company has been referred to as "one of the most influential economic and cultural forces in the world", as well as the world's most valuable brand. **Jeff Bezos PERSON** founded **Amazon ORG** from his garage in **Bellevue GPE**, **Washington GPE**, on **July 5, 1994 DATE**. It started as an online marketplace for books but expanded to sell electronics, software, video games, apparel, furniture, food, toys, and jewelry. In **2015 DATE**, **Amazon ORG** surpassed **Walmart ORG** as the most valuable retailer in the **United States GPE** by market capitalization. In **2017 DATE**, **Amazon ORG** acquired **Whole Foods Market ORG** for **US\$13.4 billion MONEY**, which substantially increased its footprint as a physical retailer. In **2016 DATE**, its **two-day DATE** delivery service, **Amazon Prime ORG**, surpassed **100 million CARDINAL** subscribers worldwide. **Amazon ORG** is known for its disruption of well-established industries through technological innovation and mass scale. It is the world's largest online marketplace, **AI ORG** assistant provider, live-streaming platform and cloud computing platform as measured by revenue and market capitalization. **Amazon ORG** is the largest Internet company by revenue in the world. It is the **second ORGINAL** largest private employer in the **United States GPE** and **one CARDINAL** of the world's most valuable companies. As of **2020 DATE**, **Amazon ORG** has the highest global brand valuation. **Amazon ORG** distributes a variety of downloadable and streaming content through its **Amazon Prime Video ORG**, **Amazon Music ORG**, **Twitch PERSON**, and **Audible ORG** subsidiaries. **Amazon ORG** also has a publishing arm, **Amazon Publishing ORG**, film and television studio **Amazon Studios ORG**, and a cloud computing subsidiary, **Amazon Web Services ORG**. It produces consumer electronics including Kindle e-readers, Fire tablets, **Fire TV ORG** and **Echo ORG** devices. Its acquisitions over the years **DATE** include **Ring ORG**, **Twitch**, **Whole Foods Market**, and **IMCn**. **Amazon ORG** is currently in the process of purchasing film and television studio **Metro-Goldwyn-Mayer ORG**. **Amazon ORG** has been criticized for practices including technological surveillance overreach, a hyper-competitive and demanding work culture, tax avoidance, and anti-competitive behavior.

```
1 #visualization
2 doc_4 = MER_4(Netflix)
3 spacy.disply.render(doc_4, style='ent', jsyter=True)
```

**Netflix, Inc. ORG** is an **American NORP** over-the-top content platform and production company headquartered in **Los Gatos GPE**, **California GPE**. **Netflix ORG** was founded in **1997 DATE** by **Reed Hastings PERSON** and **Marc Randolph PERSON** in **Scotts Valley GPE**, **California GPE**. The company's primary business is a subscription-based streaming service offering online streaming from a library of films and television series, including those produced in-house. As of **July 2021 DATE**, **Netflix ORG** had **209 million CARDINAL** subscribers, including **72 million CARDINAL** in the **United States GPE** and **Canada GPE**. It is available worldwide except in mainland **China GPE** (due to local restrictions), **Syria GPE**, **North Korea GPE** and **Cosmos GPE** (due to **US GPE** sanctions). The company has offices in **Canada GPE**, **France GPE**, **Brazil GPE**, the **Netherlands GPE**, **India GPE**, **Japan GPE**, **South Korea GPE** and the **United Kingdom GPE**. **Netflix ORG** is a member of the **Motion Picture Association ORG** (**MPA ORG**), producing and distributing content from countries all over the globe. **Netflix ORG** is initial business model included DVD sales and rental by mail, but **Hastings PERSON** abandoned the sales about a year DATE after the company's founding to focus on the initial DVD rental business. **Netflix ORG** expanded its business in **2007 DATE** with the introduction of streaming media while retaining the DVD and **Blu-ray ORG** rental business. The company expanded internationally in **2010 DATE** with streaming available in **Canada GPE**, followed by **Latin America LOC** and the **Caribbean LOC**. **Netflix ORG** entered the content-production industry in **2013 DATE** debuting its first ORGINAL series **House of Cards ORG**. Since **2012 DATE**, **Netflix ORG** has taken more of an active role as producer and distributor for both film and television series, and to that end, offers a variety of "Netflix Originals". **WORK\_OF\_ART** content through its online library. By **January 2016 DATE**, **Netflix ORG** services operated in more than **190 CARDINAL** countries. **Netflix ORG** released an estimated **126 CARDINAL** original series and films in **2016 DATE**, more than any other network or cable channel. As of **December 31, 2020 DATE**, the company had **\$16 billion MONEY** in long term debt, which it accumulated to fund its growth. The company is ranked **164th ORGINAL** on the **Fortune 500** and **PRODUCT** **254th** on the **Forbes Global 2000**. On **July 10, 2020 DATE**, **Netflix ORG** became the largest entertainment/media company by market capitalization. In **2021 DATE**, **Netflix ORG** was ranked as the **8th ORGINAL** most trusted brand globally by **Morning Consult WORK\_OF\_ART**. During the **2010s decade DATE**, **Netflix ORG** was the top-performing stock in the **S&P 500** stock market index, with a total return of **3,693% PERCENT**.

```
1 #visualization
2 doc_3 = MER_3(Apple)
3 spacy.disply.render(doc_3, style='ent', jsyter=True)
```

**Apple Inc. ORG** is an **American NORP** multinational technology company that specializes in consumer electronics, computer software, and online services. **Apple ORG** is the world's largest technology company by revenue (totaling **\$274.5 billion MONEY** in **2020 DATE**), and, since **January 2021 DATE**, the world's most valuable company. As of **2021 DATE**, **Apple ORG** is the world's **fourth ORGINAL** largest PC vendor by unit sales, and **fourth ORGINAL** largest smartphone manufacturer. It is one of the **Big Five CARDINAL** **American NORP** information technology companies, along with **Amazon ORG**, **Google ORG**, **Microsoft ORG**, and **Facebook ORG**. **Apple ORG** was founded by **Steve Jobs PERSON**, **Steve Wozniak PERSON**, and **Ronald Wayne PERSON** in **1976 DATE** to develop and sell **Wozniak PERSON** **Apple ORG** personal computer. It was incorporated by Jobs and **Wozniak PERSON** as **Apple Computer, Inc. ORG** in **1977 DATE**, and sales of its computers, including the **Apple II ORG**, grew quickly. They went public in **1980 DATE** to instant financial success. Over the next few years **DATE**, **Apple ORG** shipped new computer features including innovative graphical user interfaces, such as the original **Macintosh ORG**, announced with the critically acclaimed adven<sup>t</sup> **1984 DATE**. However, the high price of its products and limited application library caused problems, as did power struggles between executives. In **1985 DATE**, **Wozniak PERSON** departed **Apple ORG** amicably, while Jobs resigned to found **NextInc ORG**, taking some **Apple ORG** co-workers with him. As the market for personal computers expanded and evolved through the **1990s DATE**, **Apple ORG** lost considerable market share to the lower-priced dynasty of **Microsoft ORG**, **Windows PRODUCT**, **Intel ORG** PC clones. The board recruited CEO **Gil Amelio PERSON**, who prepared the struggling company for eventual success with extensive reforms, product focus, and layoffs in his **500 day DATE** tenure. In **1997 DATE**, **Gi ORG** bought **NextInc ORG** to resolve **Apple ORG** is unsuccessful OS strategy and bring back **Steve Jobs PERSON**, who replaced **Amelio PERSON** as CEO later that year **DATE**. **Apple ORG** returned to profitability under the revitalizing "Think Different" campaign, launching the **iMac ORG** and **iPod ORG**, opening a retail chain of **Apple Stores ORG** in **2001 DATE**, and acquiring numerous companies to broaden their software portfolio. In **2007 DATE**, the company launched the **iPhone PRODUCT** to critical acclaim and financial success. In **2011 DATE**, Jobs resigned as CEO due to health complications, and died **two months later DATE**. He was succeeded by **Tim Cook PERSON** in **August 2011 DATE**. **Apple ORG** became the first ORGINAL publicly traded **U.S. GPE** company to be valued at over **\$1 billion MONEY** and the first ORGINAL valued over **\$2 billion MONEY** **two years later DATE**. It has a high level of brand loyalty and is ranked as the world's most valuable brand, as of **January 2021 DATE**, there are **1.65 billion CARDINAL** **Apple ORG** products in use worldwide. However, the company receives significant criticism regarding the labor practices of its contractors, its environmental practices, and business ethics, including anti-competitive behavior, and materials sourcing.

```
1 #visualization
2 doc_5 = MER_5(Google)
3 spacy.disply.render(doc_5, style='ent', jsyter=True)
```

**Google LLC ORG** is an **American NORP** multinational technology company that specializes in Internet-related services and products, which include online advertising technologies, a search engine, cloud computing, software, and hardware. It is considered one of the **Big Five CARDINAL** companies in the **American NORP** information technology industry, along with **Amazon ORG**, **Facebook ORG**, **Apple ORG**, and **Microsoft ORG**. **Google ORG** was founded on **September 4, 1998 DATE** by **Larry Page PERSON** and **Sergey Brin PERSON** while they were **Ph.D. WORK\_OF\_ART** students at **Stanford University ORG** in **California GPE**. Together they own about **14% PERCENT** of its publicly listed shares and control **56% PERCENT** of the stockholder voting power through super-voting stock. The company went public via an initial public offering (IPO) in **2004 DATE**. In **2015 DATE**, **Google ORG** was reorganized as a wholly owned subsidiary of **Alphabet Inc. ORG**. **Google** is **Alphabet ORG**'s largest subsidiary and is a holding company for **Alphabet ORG**'s Internet properties and interests. **Sundar Pichai PERSON** was appointed CEO of **Google ORG** on **October 24, 2015 DATE**, replacing **Larry Page PERSON**, who became the CEO of **Alphabet ORG**. On **December 3, 2019 DATE**, **Pichai GPE** also became the CEO of **Alphabet ORG**. In **2021 DATE**, the **Alphabet Workers Union ORG** was founded, mainly composed of **Google ORG** employees. The company's rapid growth since incorporation has included products, acquisitions, and partnerships beyond **Google**'s core search engine, e.g., **Google Search ORG**. It offers services designed for work and productivity (Google Docs, **Google Sheets ORG**, and **Google Slides PERSON**), email ( **Gmail ORG**), scheduling and time management (Google Calendar), cloud storage (Google Drive), instant messaging and video chat (Google Duo, **Google Chat ORG**), language translation (Google Translate), mapping and navigation (Google Maps, **Vaze PERSON**, **Google Earth**, and **Street View FAC**), podcast hosting ( **Google Podcasts ORG** ), video sharing ( **YouTube ORG** ), blog publishing (Blogger), note-taking (Google Keep) and **Zoomboard PERSON**, and photo organizing and editing (Google Photos). The company leads the development of the **Android ORG** mobile operating system, the Google Chrome web browser, and Chrome OS (a lightweight, proprietary operating system based on the free and open source **Chromium ORG** OS operating system). **Google ORG** has moved increasingly into hardware, from **2010 to 2015 DATE**, it partnered with major electronics manufacturers in the production of its **Google Nexus ORG** devices, and it released multiple hardware products in **2016 DATE**, including the Google Pixel line of smartphones, **Google Home ORG** smart speaker, **Google WiM mesh wireless router ORG**, **Google ORG** has also experimented with becoming an Internet carrier ( **Google Fiber ORG** and Google Fi). **Google.com ORG** is the most visited website worldwide. Several other **Google ORG** owned websites also are on the list of most popular websites, including **YouTube ORG** and **Blogger**. On the list of most valuable brands, **Google ORG** is ranked **second ORGINAL** by **Forbes ORG** and **fourth ORGINAL** by **Interbrand ORG**. It has received significant criticism involving issues such as privacy concerns, tax avoidance, censorship, search neutrality, antitrust and abuse of its monopoly position.

Prefix	Value
faang	http://www.ftsm.ukm.my/faangOnt#
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xml	http://www.w3.org/XML/1998/namespace
xsd	http://www.w3.org/2001/XMLSchema#

Active ontology: Facebook — http://www.ftsm.ukm.my/faangOnt#Facebook

Annotations: Usage

Class hierarchy: Facebook

Asserted

owl:Thing

- BigTech
  - FAANG
    - Amazon
    - Apple
    - Facebook
    - Google
    - Netflix

Show: this disjoints named sub/superclasses

Found 40 uses of Facebook

- Facebook SubClassOf FAANG
- Class: Facebook
- FacebookCARDINAL FacebookCARDINAL Type Facebook
- FacebookDATE FacebookDATE Type Facebook

Description: Facebook

Equivalent To

SubClass Of

- FAANG

General class axioms

SubClass Of (Anonymous Ancestor)

Instances

- FacebookCARDINAL
- FacebookDATE
- FacebookEVENT
- FacebookFAC
- FacebookGPE
- FacebookLANGUAGE
- FacebookLAW
- FacebookLOC

## 3. Results and Discussion (cont'd)

### Prefixes, Class, Properties and Individuals.

Active ontology: Entities • Individuals by class • DL Query • OntoGraf • SPARQL Query

Data properties: Annotation properties: Datatypes: Individuals: Annotations: Usage

Classes: Object properties

Data property hierarchy: hasPersonName

Usage: hasPersonName

Asserted

Show: this disjoints

Found 50 uses of hasPersonName

- AmazonPERSON
  - AmazonPERSON hasPersonName "Jeff Bezos" xsd:string
  - AmazonPERSON hasPersonName "Twinkl" xsd:string
- ApplePERSON
  - ApplePERSON hasPersonName "Amelio" xsd:string
  - ApplePERSON hasPersonName "Wozniak" xsd:string
  - ApplePERSON hasPersonName "Steve Wozniak" xsd:string
  - ApplePERSON hasPersonName "Tim Cook" xsd:string

Characteristics: Description: hasPersonName

Functional

Equivalent To

SubProperty Of

Domains (Intersection)

Ranges

- xsd:string

Disjoint With

Active ontology: Entities • Individuals by class • DL Query • OntoGraf • SPARQL Query

Data properties: Annotation properties: Datatypes: Individuals: Annotations: Usage

Classes: Object properties

Individuals: FacebookPERSON

Annotations: FacebookPERSON

Description: FacebookPERSON

Types

- Facebook

Same Individual As

Different Individuals

Property assertions: FacebookPERSON

Object property assertions

- hasPersonName "Andrew McCollum" xsd:string
- hasPersonName "Mark Zuckerberg" xsd:string
- hasPersonName "Eduardo Saverin" xsd:string
- hasPersonName "Facebook Messenger" xsd:string
- hasPersonName "Destin Moskowitz" xsd:string
- hasPersonName "Chris Hughes" xsd:string

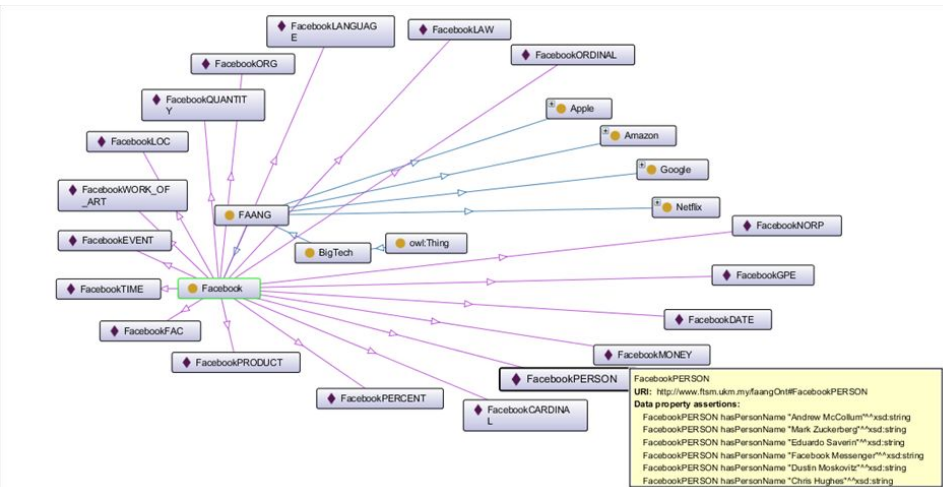
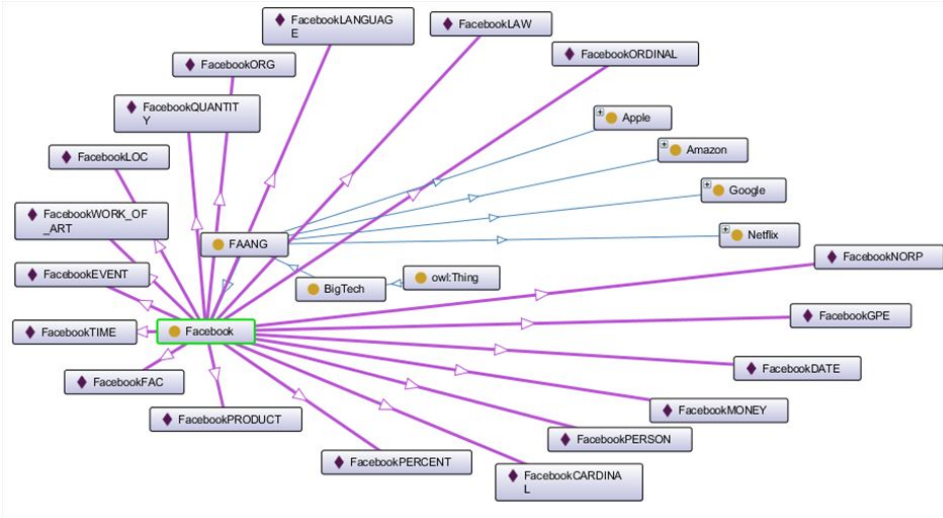
Negative object property assertions

Negative data property assertions

To use the reasoner click Reasoner • Start reasoner • Show Inferences

### 3. Results and Discussion (cont'd)

Radial hierarchy.





### 3. Results and Discussion (cont'd)

FAANG types.

SPARQL query:	
<pre>PREFIX faang: &lt;http://www.ftsm.ukm.my/faangOnt#&gt; PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; PREFIX owl: &lt;http://www.w3.org/2002/07/owl#&gt; PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt; PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema#&gt; SELECT ?subject WHERE {   ?subject rdf:type faang:FAANG }</pre>	
subject	
Google	
Apple	
Amazon	
Netflix	
Facebook	

- Q1. Who are the founder(s) of respective FAANG?
1. Mark Zuckerberg, Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes
  2. Jeff Bezos
  3. Steve Jobs, Steve Wozniak and Ronald Wayne
  4. Reed Hastings and Mark Randolph
  5. Larry Page and Sergey Brin

SPARQL query		
<pre>PREFIX faang: &lt;http://www.ftm.ukm.my/faangOnt#&gt; PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; PREFIX owl: &lt;http://www.w3.org/2002/07/owl#&gt; PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt; PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema#&gt; SELECT ?subject ?object WHERE {   ?subject rdf:type faang:Facebook .   ?subject faang:hasPersonName ?object . }</pre>		
subject		object
FacebookPERSON		"Dustin Moskovitz"^^<http://www.w3.org/2001/XMLSchema#string>
FacebookPERSON		"Eduardo Saverin"^^<http://www.w3.org/2001/XMLSchema#string>
FacebookPERSON		"Mark Zuckerberg"^^<http://www.w3.org/2001/XMLSchema#string>
FacebookPERSON		"Facebook Messenger"^^<http://www.w3.org/2001/XMLSchema#string>
FacebookPERSON		"Chris Hughes"^^<http://www.w3.org/2001/XMLSchema#string>
FacebookPERSON		"Andrew McCollum"^^<http://www.w3.org/2001/XMLSchema#string>

SPARQL query		
<pre>PREFIX faang: &lt;http://www.ftm.ukm.my/faangOnt#&gt; PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; PREFIX owl: &lt;http://www.w3.org/2002/07/owl#&gt; PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt; PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema#&gt; SELECT ?subject ?object WHERE {   ?subject rdf:type faang:Amazon .   ?subject faang:hasPersonName ?object . }</pre>		
subject		object
AmazonPERSON		"Twitch"^^<http://www.w3.org/2001/XMLSchema#string>
AmazonPERSON		"Jeff Bezos"^^<http://www.w3.org/2001/XMLSchema#string>

### 3. Results and Discussion (cont'd)

Q1 (Facebook, Amazon, Apple and Netflix).

SPARQL query		
<pre>PREFIX faang: &lt;http://www.ftm.ukm.my/faangOnt#&gt; PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; PREFIX owl: &lt;http://www.w3.org/2002/07/owl#&gt; PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt; PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema#&gt; SELECT ?subject ?object WHERE {   ?subject rdf:type faang:Apple .   ?subject faang:hasPersonName ?object . }</pre>		
subject		object
ApplePERSON		"Amelio"^^<http://www.w3.org/2001/XMLSchema#string>
ApplePERSON		"Ronald Wayne"^^<http://www.w3.org/2001/XMLSchema#string>
ApplePERSON		"Wozniak"^^<http://www.w3.org/2001/XMLSchema#string>
ApplePERSON		"Gil Amelio"^^<http://www.w3.org/2001/XMLSchema#string>
ApplePERSON		"Steve Wozniak"^^<http://www.w3.org/2001/XMLSchema#string>
ApplePERSON		"Tim Cook"^^<http://www.w3.org/2001/XMLSchema#string>
ApplePERSON		"Steve Jobs"^^<http://www.w3.org/2001/XMLSchema#string>

SPARQL query		
<pre>PREFIX faang: &lt;http://www.ftm.ukm.my/faangOnt#&gt; PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; PREFIX owl: &lt;http://www.w3.org/2002/07/owl#&gt; PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt; PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema#&gt; SELECT ?subject ?object WHERE {   ?subject rdf:type faang:Netflix .   ?subject faang:hasPersonName ?object . }</pre>		
subject		object
NetflixPERSON		"Hastings"^^<http://www.w3.org/2001/XMLSchema#string>
NetflixPERSON		"Reed Hastings"^^<http://www.w3.org/2001/XMLSchema#string>
NetflixPERSON		"Marc Randolph"^^<http://www.w3.org/2001/XMLSchema#string>

### 3. Results and Discussion (cont'd)

Q1 (Google).

SPARQL query	
<pre>PREFIX faang: &lt;http://www.ftsm.ukm.my/faangOnt#&gt; PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; PREFIX owl: &lt;http://www.w3.org/2002/07/owl#&gt; PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt; PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema#&gt; SELECT ?subject ?object WHERE {   ?subject rdf:type faang:Google .   ?subject faang:hasPersonName ?object . }</pre>	
subject	object
GooglePERSON	"Jamboard" <sup>14</sup> <http://www.w3.org/2001/XMLSchema#string>
GooglePERSON	"Sundar Pichai" <sup>14</sup> <http://www.w3.org/2001/XMLSchema#string>
GooglePERSON	"Waze" <sup>14</sup> <http://www.w3.org/2001/XMLSchema#string>
GooglePERSON	"Larry Page" <sup>14</sup> <http://www.w3.org/2001/XMLSchema#string>
GooglePERSON	"Google Slides" <sup>14</sup> <http://www.w3.org/2001/XMLSchema#string>
GooglePERSON	"Sergey Brin" <sup>14</sup> <http://www.w3.org/2001/XMLSchema#string>

Q2. Where are they founded?

- 1. US
- 2. Bellevue, Washington
- 3. US
- 4. Los Gatos, California
- 5. California

3. Results and Discussion (cont'd)

Q2 (Facebook, Amazon and Apple).

SPARQL query	
<pre>PREFIX faang: &lt;http://www.ftm.ukm.my/faangOnt#&gt; PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; PREFIX owl: &lt;http://www.w3.org/2002/07/owl#&gt; PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt; PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema#&gt; SELECT ?subject ?object WHERE {   ?subject rdf:type faang:Facebook .   ?subject faang:hasCountriesCitiesState ?object . }</pre>	
subject	object
FacebookGPE	"Cambridge Analytica" <u>http://www.w3.org/2001/XMLSchema#string</u>
FacebookGPE	"U.S." <u>http://www.w3.org/2001/XMLSchema#string</u>

SPARQL query	
<pre>PREFIX faang: &lt;http://www.ftm.ukm.my/faangOnt#&gt; PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; PREFIX owl: &lt;http://www.w3.org/2002/07/owl#&gt; PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt; PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema#&gt; SELECT ?subject ?object WHERE {   ?subject rdf:type faang:Amazon .   ?subject faang:hasCountriesCitiesState ?object . }</pre>	
subject	object
AmazonGPE	"Washington" <u>http://www.w3.org/2001/XMLSchema#string</u>
AmazonGPE	"U.S." <u>http://www.w3.org/2001/XMLSchema#string</u>
AmazonGPE	"the United States" <u>http://www.w3.org/2001/XMLSchema#string</u>
AmazonGPE	"Bellevue" <u>http://www.w3.org/2001/XMLSchema#string</u>

SPARQL query	
<pre>PREFIX faang: &lt;http://www.ftm.ukm.my/faangOnt#&gt; PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; PREFIX owl: &lt;http://www.w3.org/2002/07/owl#&gt; PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt; PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema#&gt; SELECT ?subject ?object WHERE {   ?subject rdf:type faang:Apple .   ?subject faang:hasCountriesCitiesState ?object . }</pre>	
subject	object
AppleGPE	"U.S." <u>http://www.w3.org/2001/XMLSchema#string</u>

### 3. Results and Discussion (cont'd)

Q2 (Netflix and Google).

SPARQL query:

```
PREFIX faang: <http://www.ftsm.ukm.my/faangOnt#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?subject ?object
WHERE
{
  ?subject rdf:type faang:Netflix .
  ?subject faang:hasCountriesCitiesState ?object .
}
```

subject	object
NetflixGPE	"Los Gatos""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"Japan""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"Canada""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"California""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"Crimea""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"South Korea""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"North Korea""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"India""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"the United Kingdom""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"Syria""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"China""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"Netherlands""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"Brazil""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"Scotts Valley""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"the United States""<http://www.w3.org/2001/XMLSchema#string>
NetflixGPE	"France""<http://www.w3.org/2001/XMLSchema#string>

Execute

SPARQL query:

```
PREFIX faang: <http://www.ftsm.ukm.my/faangOnt#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?subject ?object
WHERE
{
  ?subject rdf:type faang:Google .
  ?subject faang:hasCountriesCitiesState ?object .
}
```

subject	object
GoogleGPE	"Pichai""<http://www.w3.org/2001/XMLSchema#string>
GoogleGPE	"California""<http://www.w3.org/2001/XMLSchema#string>

Q3. When are they founded?

1. 2004

2. July 5, 1994

3. 1976

4. 1997

5. September 4, 1998

SPARQL query:

```
PREFIX faang: <http://www.ftsm.ukm.my/faangOnt#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?subject ?object
WHERE
{
  ?subject rdf:type faang:Facebook .
  ?subject faang:hasDate ?object .
}
```

subject	object
FacebookDATE	"2006"<^^<http://www.w3.org/2001/XMLSchema#string>
FacebookDATE	"13 years old"<^^<http://www.w3.org/2001/XMLSchema#string>
FacebookDATE	"2016"<^^<http://www.w3.org/2001/XMLSchema#string>
FacebookDATE	"2004"<^^<http://www.w3.org/2001/XMLSchema#string>
FacebookDATE	"2006"<^^<http://www.w3.org/2001/XMLSchema#string>

SPARQL query:

```
PREFIX faang: <http://www.ftsm.ukm.my/faangOnt#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?subject ?object
WHERE
{
  ?subject rdf:type faang:Amazon .
  ?subject faang:hasDate ?object .
}
```

subject	object
AmazonDATE	"July 5, 1994"<^^<http://www.w3.org/2001/XMLSchema#string>
AmazonDATE	"2018"<^^<http://www.w3.org/2001/XMLSchema#string>
AmazonDATE	"2017"<^^<http://www.w3.org/2001/XMLSchema#string>
AmazonDATE	"the years"<^^<http://www.w3.org/2001/XMLSchema#string>
AmazonDATE	"2015"<^^<http://www.w3.org/2001/XMLSchema#string>
AmazonDATE	"Two-day"<^^<http://www.w3.org/2001/XMLSchema#string>
AmazonDATE	"2020"<^^<http://www.w3.org/2001/XMLSchema#string>

## 3. Results and Discussion (cont'd)

Q3 (Facebook, Amazon and Apple).

SPARQL query:

```
PREFIX faang: <http://www.ftsm.ukm.my/faangOnt#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?subject ?object
WHERE
{
  ?subject rdf:type faang:Apple .
  ?subject faang:hasDate ?object .
}
```

subject	object
AppleDATE	"2020"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"2021"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"2007"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"1976"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"two years later"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"1977"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"the 1990s"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"the next few years"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"500 day"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"1984"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"January 2021"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"August 2018"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"1985"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"1997"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"later that year"<^^<http://www.w3.org/2001/XMLSchema#string>
AppleDATE	"1980"<^^<http://www.w3.org/2001/XMLSchema#string>

### 3. Results and Discussion (cont'd)

Q3 (Netflix and Google).

SPARQL query	
<pre>PREFIX faang: &lt;http://www.ftm.ukm.my/faangOnt#&gt; PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; PREFIX owl: &lt;http://www.w3.org/2002/07/owl#&gt; PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt; PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema#&gt; SELECT ?subject ?object WHERE {   ?subject rdf:type faang:Netflix .   ?subject faang:hasDate ?object . }</pre>	
subject	object
NetflixDATE	"July 2021"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"2010"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"2021"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"July 10, 2020"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"2013"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"2012"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"1997"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"2016"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"about a year"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"December 31, 2020"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"the 2010s decade"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"January 2016"<<http://www.w3.org/2001/XMLSchema#string>
NetflixDATE	"2007"<<http://www.w3.org/2001/XMLSchema#string>

SPARQL query	
<pre>PREFIX faang: &lt;http://www.ftm.ukm.my/faangOnt#&gt; PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; PREFIX owl: &lt;http://www.w3.org/2002/07/owl#&gt; PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt; PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema#&gt; SELECT ?subject ?object WHERE {   ?subject rdf:type faang:Google .   ?subject faang:hasDate ?object . }</pre>	
subject	object
GoogleDATE	"2015"<<http://www.w3.org/2001/XMLSchema#string>
GoogleDATE	"2010 to 2015"<<http://www.w3.org/2001/XMLSchema#string>
GoogleDATE	"2004"<<http://www.w3.org/2001/XMLSchema#string>
GoogleDATE	"2016"<<http://www.w3.org/2001/XMLSchema#string>
GoogleDATE	"2021"<<http://www.w3.org/2001/XMLSchema#string>
GoogleDATE	"October 24, 2015"<<http://www.w3.org/2001/XMLSchema#string>
GoogleDATE	"December 3, 2019"<<http://www.w3.org/2001/XMLSchema#string>
GoogleDATE	"September 4, 1998"<<http://www.w3.org/2001/XMLSchema#string>



### 3. Results and Discussion (cont'd)

#### Q4 (Facebook and Amazon)

**Table 10. Q4 Answer versus SPARQL for Amazon**

Q4 Answer	No.	SPARQL (hasCompaniesAgenciesInstitutions)
Amazon	1	Amazon
Whole Foods Market	2	Whole Foods Market
Amazon Prime Video	3	Amazon Prime Video
Amazon Music	4	Amazon Music
Twitch	5	(hasPersonName) Twitch
Audible	6	Audible
Amazon Publishing	7	Amazon Publishing
Amazon Studios	8	Amazon Studios
Amazon Web Services	9	Amazon Web Services
Kindle	10	-NA-
Fire tablets	11	-NA-
Fire TV	12	Fire TV
Echo	13	Echo
Ring	14	Ring
IMDb	15	-NA-
15	Count	12

Q4 is the competency question where there are 3 scenarios namely exact match, partial match and no match, exact match refers to queries that requires only a single predicate for matching respective FAANG's competency question while partial match refers to queries that requires multiple predicates for matching respective FAANG's competency question.

For Facebook and Netflix as shown in Table 9 and 12 respectively, the SPARQL query returns an exact match where it only requires predicate to match Q4's answers while for Apple the SPARQL query returns a partial match 2 different predicates. Amazon and Google both contains a mixture of partial match and no match.

It can be seen that fine-tuning is required for better identification of "PRODUCT" because "hasProduct" if used as the predicate will only return an iPhone as can be seen in Table 11, most of the predicate that returns answers matching the competency questions comes from "hasCompaniesAgenciesInstitutions" which identifies as an "ORG" or company. Some comes from "hasPersonName" that identifies as a "PERSON" in the case of Amazon and Google as can be seen in Table 10 and 13 respectively. In the case of Google as well, "hasBuildingsAirportsHighwaysBridges" that identifies as "FAC" or facility was observed. For companies with many products such as Amazon and Google, there are numerous no matches "-NA-".

**Table 9. Q4 Answer versus SPARQL for Facebook**

Q4 Answer	No.	SPARQL (hasCompaniesAgenciesInstitutions)
Facebook	1	Facebook



### 3. Results and Discussion (cont'd)

#### Q4 (Apple, Netflix and Google)

**Table 11.** Q4 Answer versus SPARQL for Apple

Q4 Answer	No.	SPARQL (hasCompaniesAgenciesInstitutions)
iMac	1	iMac
iPod	2	iPod
iPhone	3	(hasProduct) iPhone

**Table 12.** Q4 Answer versus SPARQL for Netflix

Q4 Answer	No.	SPARQL (hasCompaniesAgenciesInstitutions)
Netflix	1	Netflix

**Table 13.** Q4 Answer versus SPARQL for Google

Q4 Answer	No.	SPARQL (hasCompaniesAgenciesInstitutions)
Google Search	1	Google Search
Google Docs	2	-NA-
Google Sheets	3	Google Sheets
Google Slides	4	(hasPersonName) Google Slides
Gmail	5	Gmail
Google Calendar	6	-NA-
Google Drive	7	-NA-
Google Duo	8	-NA-
Google Chat	9	Google Chat
Google Meet	10	-NA-
Google Translate	11	-NA-
Google Maps	12	-NA-
Waze	13	(hasPersonName) Waze
Google Earth	14	-NA-
Street View	15	(hasBuildingsAirportsHighwaysBridges) Street View
Google Podcasts	16	Google Podcasts
YouTube	17	YouTube
Blogger	18	-NA-
Google Keep	19	-NA-
Jamboard	20	(hasPersonName) Jamboard
Google Photos	21	-NA-
Google Chrome	22	-NA-
Android	23	Android
Chromium OS	24	Chromium
Google Nexus	25	Nexus
Google Pixel	26	-NA-
Google Home	27	Google Home
Google Wifi	28	Google Wifi mesh wireless router
Google Fiber	29	Google Fiber
Google Fi	30	-NA-
30	Count	16

### 3. Results and Discussion (cont'd)

Q5 (Facebook, Amazon, Apple, Netflix and Google).

Q5. What is the nationality of the company? (NORP)

1. American
2. American
3. American
4. American
5. American

SPARQL query:

```
PREFIX faang: <http://www.ftm.ukm.my/faangOnt#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?subject ?object
WHERE
{
  ?subject faang:hasNationalityReligiousPolitical ?object .
}
```

subject	object
AppleNORP	"American"<^^http://www.w3.org/2001/XMLSchema#string
NetflixNORP	"American"<^^http://www.w3.org/2001/XMLSchema#string
FacebookNORP	"North American"<^^http://www.w3.org/2001/XMLSchema#string
GoogleNORP	"American"<^^http://www.w3.org/2001/XMLSchema#string
AmazonNORP	"American"<^^http://www.w3.org/2001/XMLSchema#string
FacebookNORP	"American"<^^http://www.w3.org/2001/XMLSchema#string

## 4. Conclusion

In conclusion, an **out-of-the-box pre-trained model** can be used for general Named Entity Extraction as demonstrated and further **transformed into RDF/RDFS** format for visualization in Protégé. This was done for Big Tech data consisting of FAANG companies and can be considered a **general domain**, as such also shows that it can be **applied to other domains** as well **without fine-tuning to suit that domain**. Future works, should be done using an **ensemble** of SOTA NER before NEE for better scores with **micro fine-tuning** and experiment with **k-shot learning** as experimented by Van Hoang et al. (2021).



*The End*