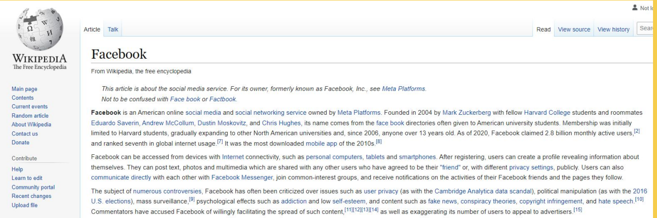
Named Entity Extraction in RDF/RDFS for Big Tech

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.

Pipeline



Not logged in

Article Talk Read View source View history Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Facebook

From Wikipedia, the free encyclopedia

This article is about the social media service. For its owner, formerly known as Facebook, Inc., see Meta Platforms.

Not to be confused with Face book or Factbook.

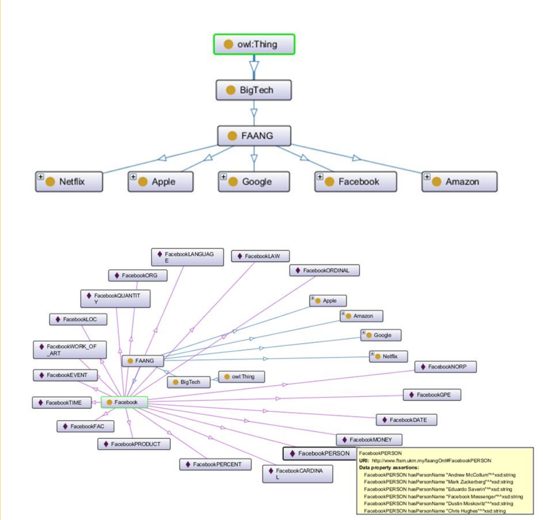
Facebook is an American online social media and social networking service owned by Meta Platforms. Founded in 2004 by Mark Zuckerberg with fellow Harvard College students and roommates Eduardo Saverin, Andrew McCollum, Dustin Moskovitz, and Chris Hughes, its name comes from the face book directories often given to American university students. Membership was initially limited to Harvard students, gradually expanding to other North American universities and, since 2006, anyone over 13 years old. As of 2020, Facebook claimed 2.8 billion monthly active users,[2] and ranked seventh in global internet usage.[7] It was the most downloaded mobile app of the 2010s.[8]

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 with Facebook Messenger, join common-interest groups, and receive notifications on the activities of their Facebook friends and the pages they follow.

The subject of numerous controversies, Facebook has often been criticized over issues such as user privacy (as with the Cambridge Analytica data scandal), political manipulation (as with the 2016 U.S. elections), mass surveillance,[9] psychological effects such as addiction and low self-esteem, and content such as fake news, conspiracy theories, copyright infringement, and hate speech.[10] Commentators have accused Facebook of willingly facilitating the spread of such content,[11][12][13][14] as well as exaggerating its number of users to appeal to advertisers.[15]

https://en.wikipedia.org/wiki/Facebook

https://www.pngkit.com/view/u2q8w7w7r5i1a9q8_pipe-8-bit-mario/

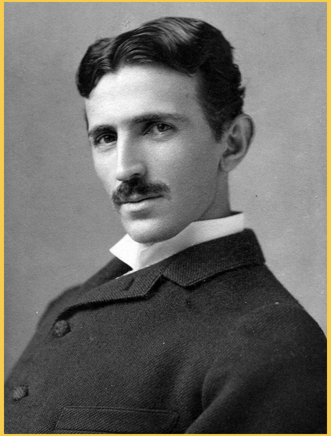


Abstract

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.

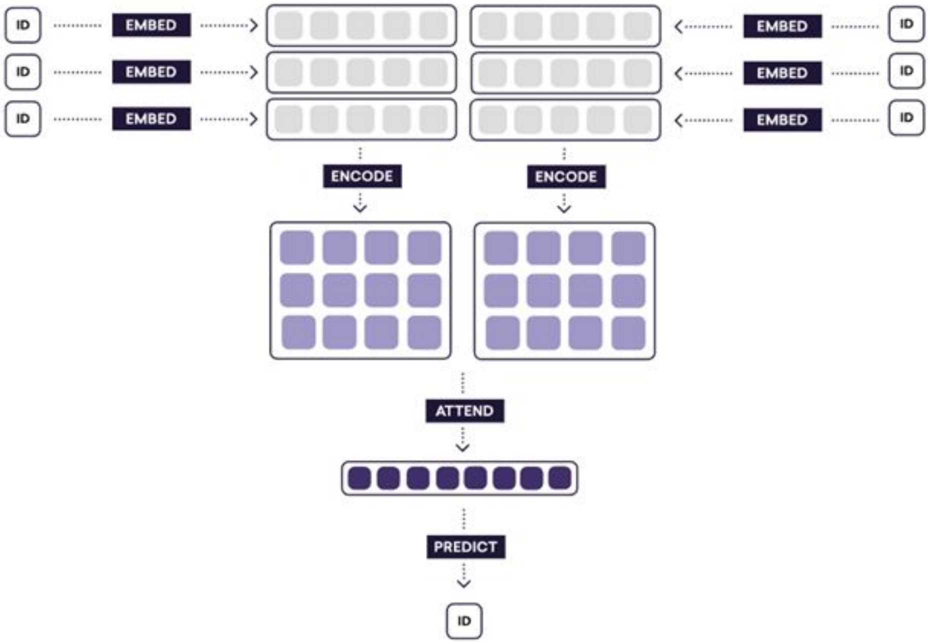


1. Introduction (cont'd)

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.

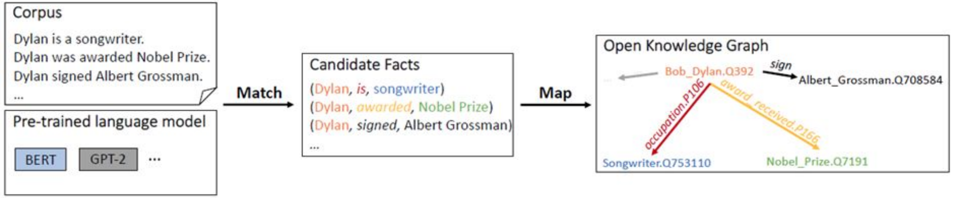
In this study, NEE was used for taxonomy extraction mainly for co-hyponyms or Abox for individual tech companies with highest frequency hypernym extracted from the pipeline. This study employed a modular approach whereby additional input data can be plugged in to further populate the taxonomy.

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.

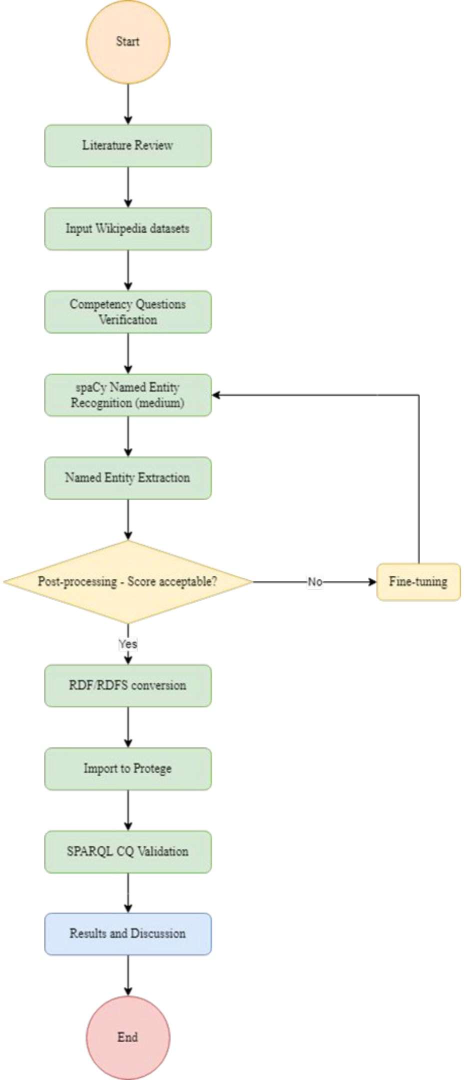


1. Introduction (cont'd)

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

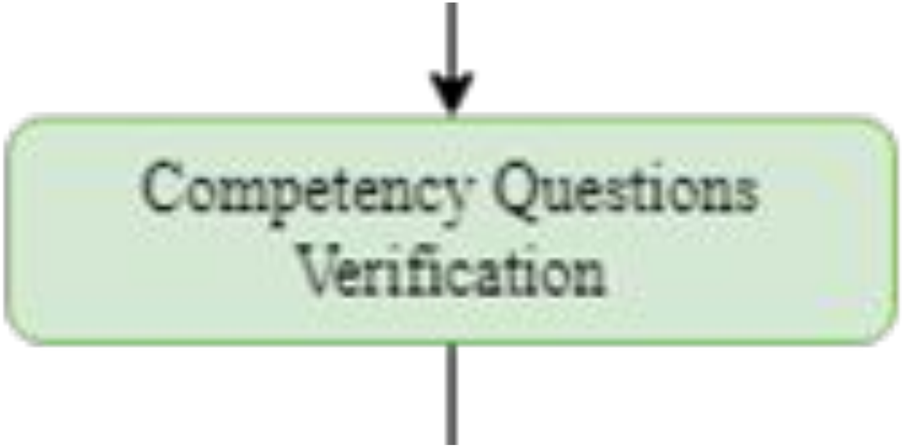
1. Literature Review.
2. Input Wikipedia datasets.
3. CQ Verification.
4. NER.
5. NEE.
6. Scoring.
7. RDF/RDFS translation.
8. Protege import.
9. SPARQL CQ Validation.
10. Results and Discussion.
11. End.

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.

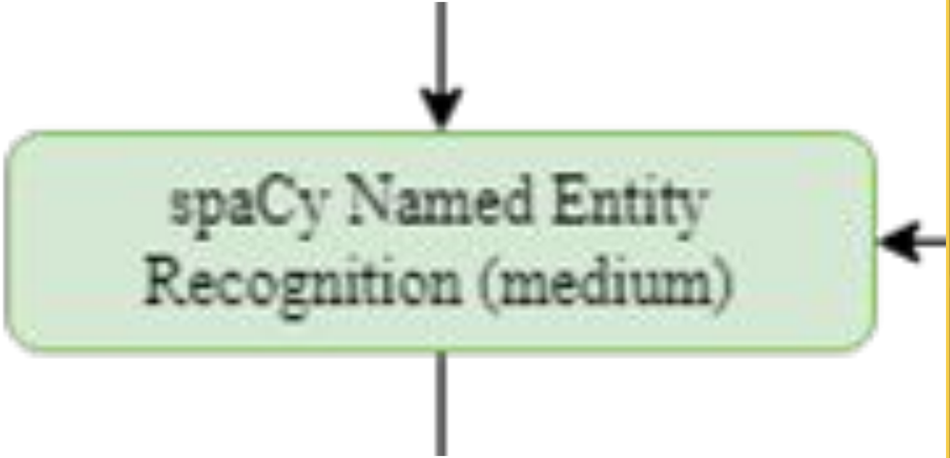
Open and Load

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



2.2 CQ 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?

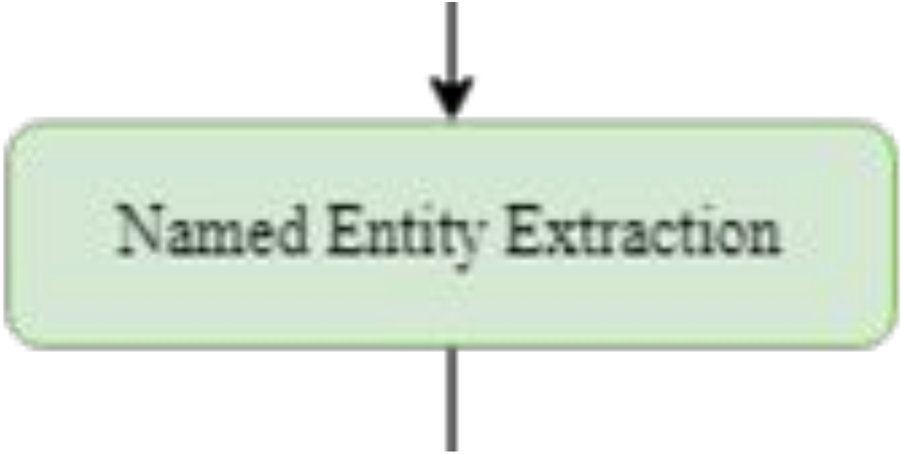


2.3 spaCy NER

```
[ ]  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)
```



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 taxanomy.

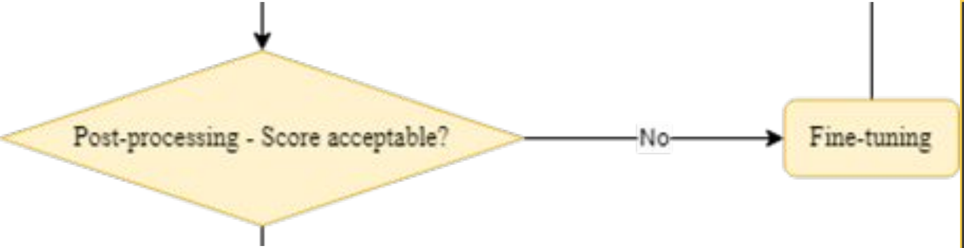
spaCy and displaCy

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

Facebookentities = []
Facebooklabels = []
Facebookstartpos = []
Facebookendpos = []

for ent in Facebookdoc.ents:
  Facebookentities.append(ent)
  Facebooklabels.append(ent.label_)
  Facebookstartpos.append(ent.start_char)
  Facebookendpos.append(ent.end_char)

Facebookdf = pd.DataFrame({'Entities':Facebookentities,'Labels':Facebooklabels, 'Position_Start':Facebookstartpos, 'Position_End':Facebookendpos})

Facebookdf

Facebookdf.to_csv(r'/content/drive/MyDrive/Colab Notebooks/OKR_NEE_datasets/out.csv', index = False)
```

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

```
[ ]
y_1[0][0] #highest frequency

'Facebook'
```

```
[ ]
#visualization
doc_1 = NER_1(Facebook)
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

```
[ ]
#equate
nlp_1 = NER_1
```

```
[ ]
#working code

Facebookexamples = []

data_1 = [(Facebook, {"entities": [(0,8,'ORG'), (15,23,'NORP'), (83,97,'ORG'), (109,113,'DATE'), (117,132,'PERSON'),
(145,160,'ORG'), (184,199,'PERSON'), (201,216,'PERSON'), (218,234,'PERSON'), (240,252,'PERSON'),
(315,323,'NORP'), (381,388,'ORG'), (428,442,'NORP'), (467,471,'DATE'), (480,497,'DATE'),
(505,509,'DATE'), (511,519,'ORG'), (528,539,'CARDINAL'), (540,547,'DATE'), (573,580,'ORDINAL'),
(648,657,'DATE'), (659,667,'ORG'), (1085,1103,'PRODUCT'), (1187,1195,'ORG'), (1266,1274,'ORG'),
(1347,1366,'ORG'), (1418,1422,'DATE'), (1423,1427,'GPE'), (1639,1647,'ORG')]})]

for text_1, annots_1 in data_1:
    doc_1 = nlp_1.make_doc(text_1)
    Facebookexamples.append(Example.from_dict(doc_1, annots_1))

#output to txt
f = sys.stdout
f = open('score_output.txt', 'a') #append to opened common file

print("#Facebook score .", file=f)
print(nlp_1.evaluate(Facebookexamples), file=f) #Global and local entity metrics
f.close() #close file
```

```
[ ]
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

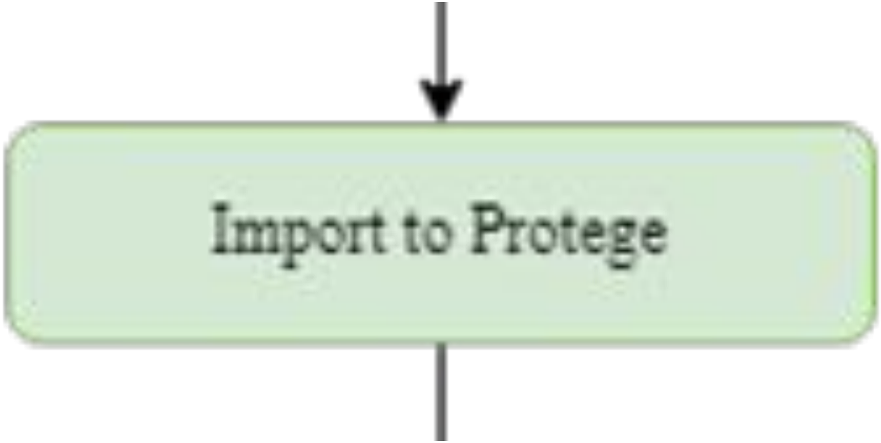
Basically, the general idea is to transform for individual 18 numbers of spaCy labels and extracted entities to be RDF/RDFS compatible.

```
#For Facebook

#output to owl
f = sys.stdout
f = open('p107443_output.owl', 'a') #append to opened common file

PropLabel_1 = y_1[0][0]

#General Prefixes
print("#Prefixes", file=f)
print("@prefix faang: <http://www.ftsm.ukm.my/faangOnt#> .", file=f)
print("@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .", file=f)
print("@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .", file=f)
print("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .", file=f)

#General Class and SubClass
print("\n#Classes and SubClasses", file=f)
print("faang:BigTech rdf:type rdfs:Class .", file=f) #Class
print("faang:FAANG rdf:type rdfs:Class .", file=f) #Class
print("faang:FAANG rdfs:subClassOf faang:BigTech .", file=f) #SubClass
print("faang:{proplab_1} rdf:type rdfs:Class .".format(proplab_1=PropLabel_1), file=f) #Class
print("faang:{proplab_1} rdfs:subClassOf faang:FAANG .".format(proplab_1=PropLabel_1), file=f) #Facebook SubClass

#Facebook Individuals
print("\n#Facebook Individuals", file=f)
print("faang:{proplab_1}PERSON rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
print("faang:{proplab_1}NORP rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
print("faang:{proplab_1}FAC rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
print("faang:{proplab_1}ORG rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
print("faang:{proplab_1}GPE rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
print("faang:{proplab_1}LOC rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
print("faang:{proplab_1}PRODUCT rdf:type faang:{proplab_1} .".format(proplab_1=PropLabel_1), file=f)
```

2.6 RDF/RDFS conversion (cont'd)

```python
#General spaCy labels to Data Properties
print("\n#spaCy labels", file=f)
PERSON = "hasPersonName" #1
NORP = "hasNationalityReligiousPolitical" #2
FAC = "hasBuildingsAirportsHighwaysBridges" #3
ORG = "hasCompaniesAgenciesInstitutions" #4
GPE = "hasCountriesCitiesState" #5
LOC = "hasLocation" #6
PRODUCT = "hasProduct" #7
EVENT = "hasEvent" #8
WORK_OF_ART = "hasWorkOfArt" #9
LAW = "hasLaw" #10
LANGUAGE = "hasLanguage" #11
DATE = "hasDate" #12
TIME = "hasTime" #13
PERCENT = "hasPercent" #14
MONEY = "hasMoney" #15
QUANTITY = "hasQuantity" #16
ORDINAL = "hasOrdinal" #17
CARDINAL = "hasCardinal" #18

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

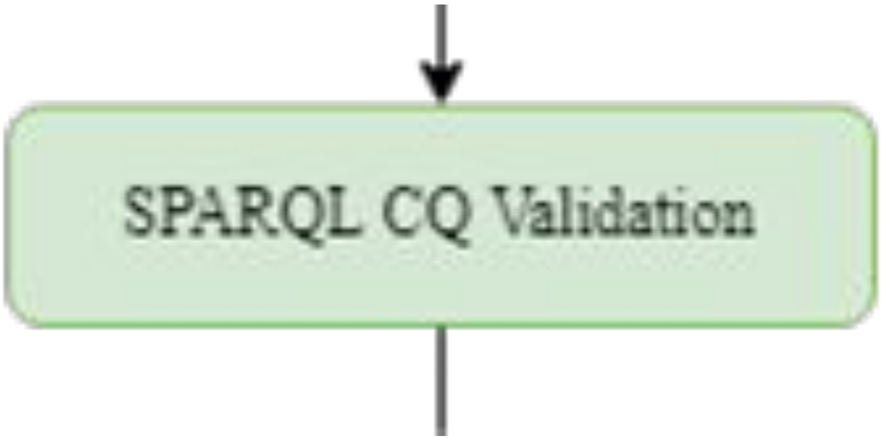
```python
#Specific Facebook Triples
print("\n#Facebook Triples", file=f)
for i in range(len(FacebooklistEntities)):
  if FacebooklistLabels[i]=='PERSON':
    print('faang:{proplab_1}PERSON faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasPersonName', ent=FacebooklistEntities[i]), file=f)
  elif FacebooklistLabels[i]=='NORP':
    print('faang:{proplab_1}NORP faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasNationalityReligiousPolitical', ent=FacebooklistEntities[i])
  elif FacebooklistLabels[i]=='FAC':
    print('faang:{proplab_1}FAC faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasBuildingsAirportsHighwaysBridges', ent=FacebooklistEntities[i
  elif FacebooklistLabels[i]=='ORG':
    print('faang:{proplab_1}ORG faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasCompaniesAgenciesInstitutions', ent=FacebooklistEntities[i]),
  elif FacebooklistLabels[i]=='GPE':
    print('faang:{proplab_1}GPE faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasCountriesCitiesState', ent=FacebooklistEntities[i]), file=f)
  elif FacebooklistLabels[i]=='LOC':
    print('faang:{proplab_1}LOC faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasLocation', ent=FacebooklistEntities[i]), file=f)
  elif FacebooklistLabels[i]=='PRODUCT':
    print('faang:{proplab_1}PRODUCT faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasProduct', ent=FacebooklistEntities[i]), file=f)
  elif FacebooklistLabels[i]=='EVENT':
    print('faang:{proplab_1}EVENT faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasEvent', ent=FacebooklistEntities[i]), file=f)
  elif FacebooklistLabels[i]=='WORK_OF_ART':
    print('faang:{proplab_1}WORK_OF_ART faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasWorkOfArt', ent=FacebooklistEntities[i]), file=f)
  elif FacebooklistLabels[i]=='LAW':
    print('faang:{proplab_1}LAW faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasLaw', ent=FacebooklistEntities[i]), file=f)
  elif FacebooklistLabels[i]=='LANGUAGE':
    print('faang:{proplab_1}LANGUAGE faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasLanguage', ent=FacebooklistEntities[i]), file=f)
  elif FacebooklistLabels[i]=='DATE':
    print('faang:{proplab_1}DATE faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasDate', ent=FacebooklistEntities[i]), file=f)
  elif FacebooklistLabels[i]=='TIME':
    print('faang:{proplab_1}TIME faang:{lab} "{ent}"^^xsd:string .' .format(proplab_1=PropLabel_1, lab='hasTime', ent=FacebooklistEntities[i]), file=f)
```



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.

```
#For Facebook

#output to owl
f = sys.stdout
f = open('p107443_output.owl', 'a') #append to opened common file
```



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
Google	**0.730159**	**0.638889**	**0.681481**
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
ORG	**0.833333**	**0.500000**	**0.625000**
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
GPE	**0.500000**	**1.000000**	**0.666667**
PRODUCT	**0.000000**	**0.000000**	**0.000000**

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
CARDINAL	**0.333333**	**1.000000**	**0.500000**
GPE	1.000000	1.000000	1.000000
PERSON	**0.500000**	**1.000000**	**0.666667**
DATE	1.000000	1.000000	1.000000
MONEY	1.000000	1.000000	1.000000
ORDINAL	1.000000	1.000000	1.000000
PRODUCT	**0.000000**	**0.000000**	**0.000000**

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
MONEY	**0.666667**	**0.666667**	**0.666667**
DATE	1.000000	1.000000	1.000000
ORDINAL	1.000000	1.000000	1.000000
CARDINAL	**0.500000**	**1.000000**	**0.666667**
PERSON	0.900000	0.692308	0.782609
PRODUCT	**1.000000**	**0.333333**	**0.500000**
GPE	1.000000	1.000000	1.000000

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 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
WORK_OF_ART	**0.000000**	**0.000000**	**0.000000**
MONEY	1.000000	1.000000	1.000000
PRODUCT	**0.000000**	**0.000000**	**0.000000**
PERCENT	1.000000	1.000000	1.000000

Table 8. Google Precision, Recall and F-score

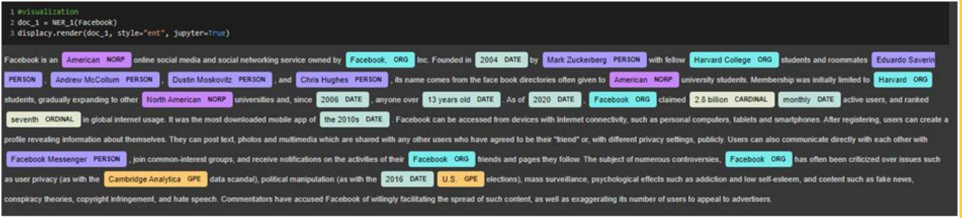
Labels	Precision	Recall	F-score
ORG	0.729730	0.794118	0.760563
NORP	1.000000	1.000000	1.000000
CARDINAL	**0.000000**	**0.000000**	**0.000000**
DATE	1.000000	1.000000	1.000000
PERSON	**0.571429**	**0.800000**	**0.666667**
WORK_OF_ART	**0.000000**	**0.000000**	**0.000000**
GPE	**0.500000**	**1.000000**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

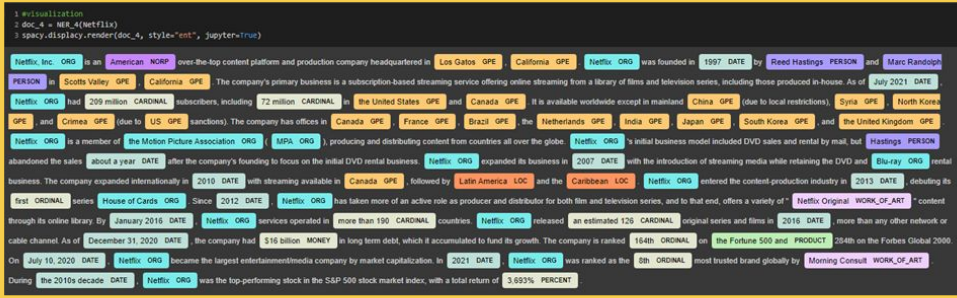
3. Results and Discussion (cont'd)

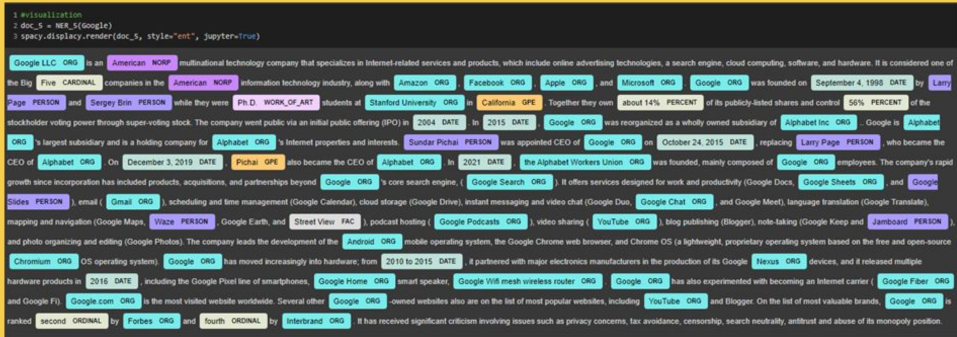
Tables 7 and 8 shows the individual labels local scores for Netflix and Google. Both of them have similar issues with "PRODUCT" label.

3. Results and Discussion (cont'd)

displaCy visualization.





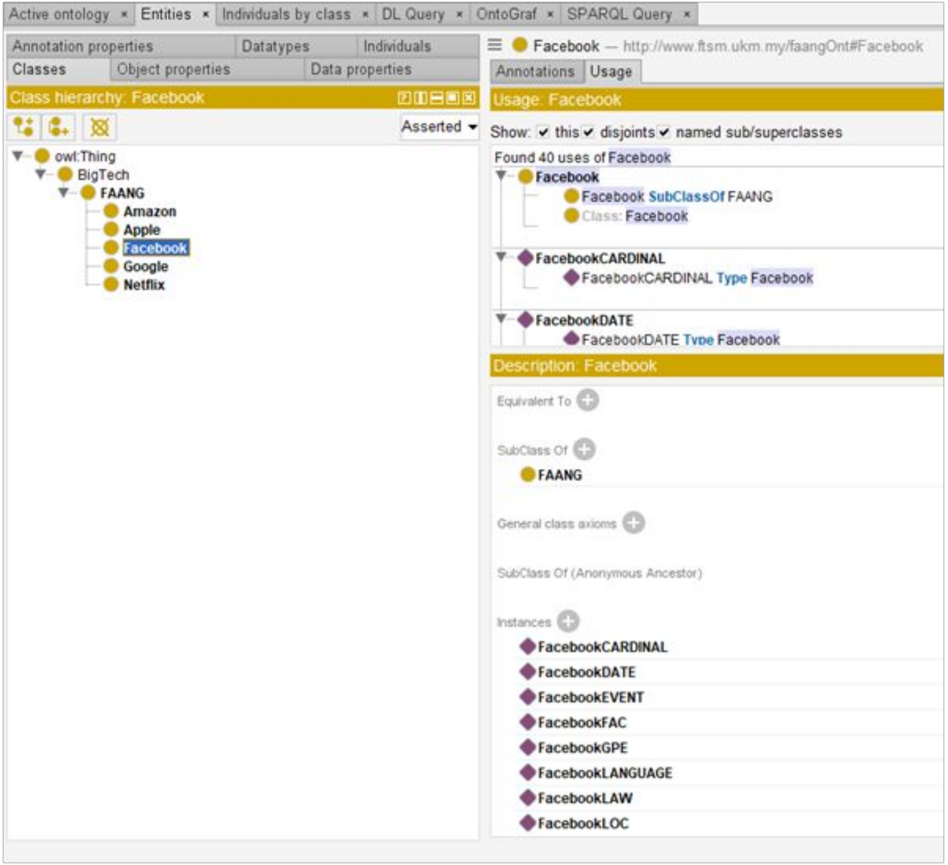


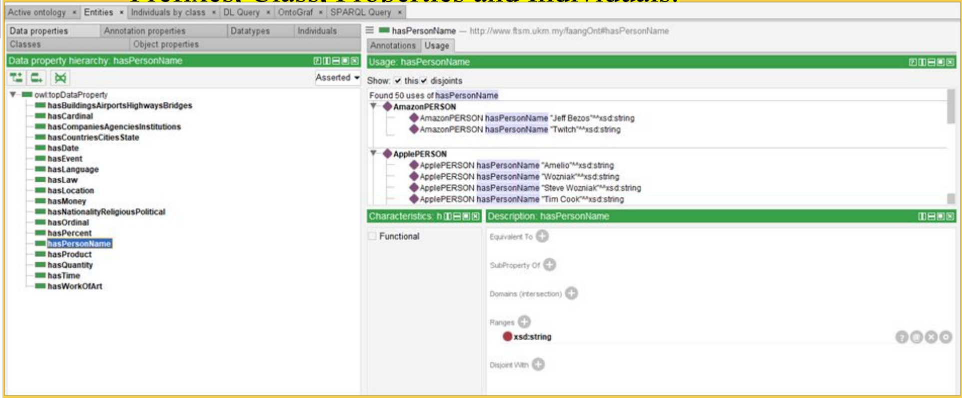
3. Results and Discussion (cont'd)

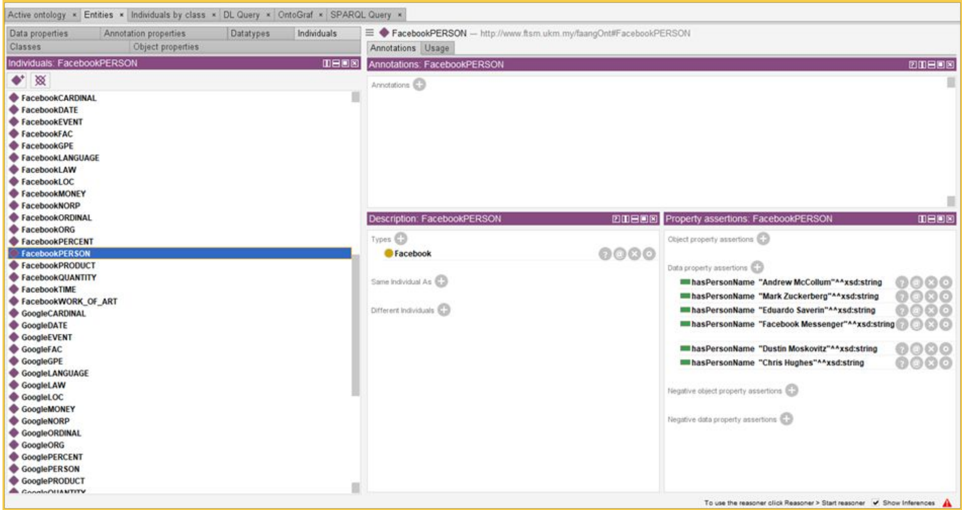
Prefixes, Class, Properties and Individuals.

Ontology prefixes:

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#

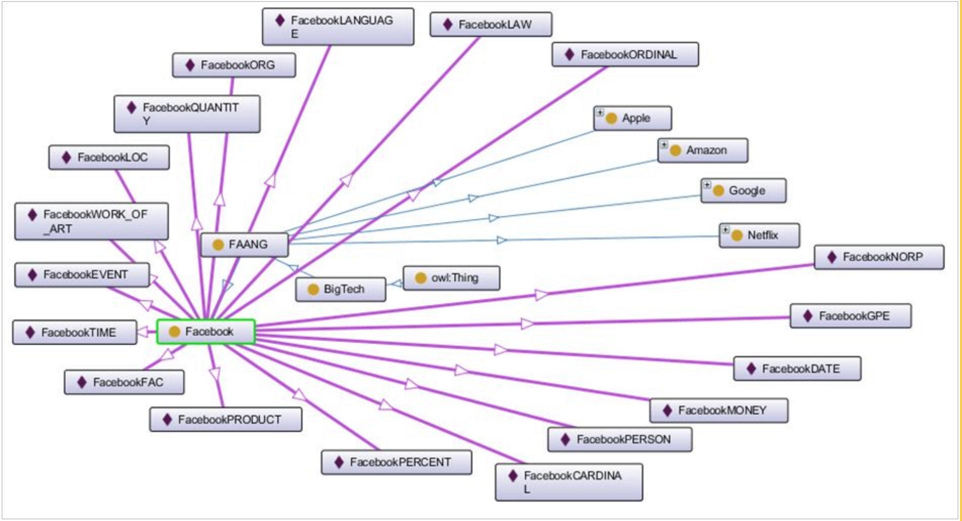


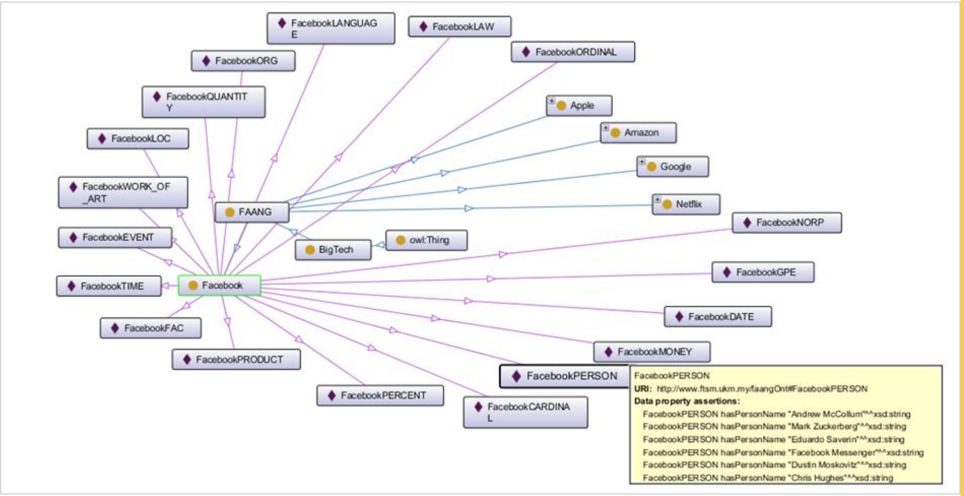




3. Results and Discussion (cont’d)

Radial hierarchy.





3. Results and Discussion (cont'd)

FAANG types.

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
WHERE
{
?subject rdf:type faang:FAANG
}
```

subject
Google
Apple
Amazon
Netflix
Facebook

3. Results and Discussion (cont'd)

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

Q1 (Facebook, Amazon, Apple and Netflix).

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:hasPersonName ?object .
}
```

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:

```
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:hasPersonName ?object .
}
```

subject	object
AmazonPERSON	"Twitch"^^<http://www.w3.org/2001/XMLSchema#string>
AmazonPERSON	"Jeff Bezos"^^<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:Apple .
?subject faang:hasPersonName ?object .
}
```

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:

```
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:hasPersonName ?object .
}
```

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:

```
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:hasPersonName ?object .
}
```

subject	object
GooglePERSON	"Jamboard"^^<http://www.w3.org/2001/XMLSchema#string>
GooglePERSON	"Sundar Pichai"^^<http://www.w3.org/2001/XMLSchema#string>
GooglePERSON	"Waze"^^<http://www.w3.org/2001/XMLSchema#string>
GooglePERSON	"Larry Page"^^<http://www.w3.org/2001/XMLSchema#string>
GooglePERSON	"Google Slides"^^<http://www.w3.org/2001/XMLSchema#string>
GooglePERSON	"Sergey Brin"^^<http://www.w3.org/2001/XMLSchema#string>

3. Results and Discussion (cont’d)

Q2 (Facebook, Amazon and Apple).

Q2. Where are they founded?

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

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:hasCountriesCitiesState ?object .
}
```

subject	object
FacebookGPE	"Cambridge Analytica"^^<http://www.w3.org/2001/XMLSchema#string>
FacebookGPE	"U.S."^^<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:hasCountriesCitiesState ?object .
}
```

subject	object
AmazonGPE	"Washington"^^<http://www.w3.org/2001/XMLSchema#string>
AmazonGPE	"U.S."^^<http://www.w3.org/2001/XMLSchema#string>
AmazonGPE	"the United States"^^<http://www.w3.org/2001/XMLSchema#string>
AmazonGPE	"Bellevue"^^<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:Apple .
?subject faang:hasCountriesCitiesState ?object .
}
```

subject	object
AppleGPE	"U.S."^^<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: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>

3. Results and Discussion (cont'd)

Q2 (Netflix and Google).

3. Results and Discussion (cont'd)

Q3 (Facebook, Amazon and Apple).

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	"monthly"^^<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>

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:

```
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:hasDate ?object .
}
```

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:

```
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:hasDate ?object .
}
```

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)

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

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

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.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 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).

https://upload.wikimedia.org/wikipedia/commons/c/cd/The-end-folks.jpg