In [ ]:

```
####### Step 1 ################ -   Importing
```

In [1]:

```
from SPARQLWrapper import SPARQLWrapper, JSON , XML
```

In [2]:

```
from eventregistry import *
import json, os, sys
```

In [3]:

```
#er = EventRegistry(allowUseOfArchive=False)
er = EventRegistry(apiKey = "63a4fe09-615a-4969-9713-6dfb630e2828")
```

```
using user provided API key for making requests
Event Registry host: http://eventregistry.org
Text analytics host: http://analytics.eventregistry.org
```

In [ ]:

```

```

In [ ]:

```
####### Step 2 ################ -   Wrapper
```

In [4]:

```
sparql = SPARQLWrapper("http://eventkginterface.l3s.uni-hannover.de/sparql")
```

In [ ]:

```

```

In [5]:

```
####### Step 3 ################ -   Enter Query
```

In [58]:

```
sparql_query = input("Please enter the sparql query :")
```

```
Please enter the sparql query :PREFIX eventKG-s: <http://eventKG.l3s.uni-hannover.de/schema/> PREFIX ev
entKG-g: <http://eventKG.l3s.uni-hannover.de/graph/> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-synt
ax-ns#> PREFIX so: <http://schema.org/> PREFIX sem: <http://semanticweb.cs.vu.nl/2009/11/sem/> PREFIX o
wl: <http://www.w3.org/2002/07/owl#> PREFIX dbr: <http://dbpedia.org/resource/>  SELECT ?place ?date WH
ERE {  ?event rdf:type sem:Event .  ?event owl:sameAs dbr:Battle_of_Trafalgar .  GRAPH eventKG-g:event_
kg {?event sem:hasPlace ?loc} .  GRAPH eventKG-g:dbpedia_en { ?loc owl:sameAs ?location . } }
```

In [ ]:

```
```

In [ ]:

```
####### Step 4 ################ -   Preprocessing and Mapping
```

In [59]:

```
def preprocessing(sparql_query):

    keyword = re.findall(r"SELECT(.+?)WHERE" , sparql_query)
    keyword_join = "".join(keyword)
    keyword_split = str(keyword_join).split()
    keyword_list = []
    for each in keyword_split:
        if each is not None:
            each = each.replace("?", "")
            keyword_list.append(each)
    return keyword_list
```

In [60]:

```
keyword_list = preprocessing(sparql_query)
keyword_list
```

Out[60]:

```
['place', 'date']
```

In [61]:

```
def event_kg_mapper(keyword_list):

    tablemap_computer = {
    'date' : "startdate",
    'place' : 'location',
    'name' : 'nomen',
    'nomen' : 'name'
    }
    tablemap = []
    for each in list(keyword_list):

        if tablemap_computer.get(each) is not None:

            tablemap.append(tablemap_computer.get(each))

    return tablemap
```

In [64]:

```
def event_registry_mapper(keyword_list):

    tablemap_computer = {
    'date' : "eventDate",
    'place' : 'location',
    'name' : 'naam',

    }
    tablemap = []
    for each in list(keyword_list):

        if tablemap_computer.get(each) is not None:

            tablemap.append(tablemap_computer.get(each))

    return tablemap
```

In [65]:

```
event_kg_mapped_keyword_list = event_kg_mapper(keyword_list)
print(event_kg_mapped_keyword_list)
event_registry_mapped_keyword_list = event_registry_mapper(keyword_list)
print(event_registry_mapped_keyword_list)
```

```
['location', 'startdate']
['location', 'eventDate']
```

In [ ]:

In [ ]:

In [ ]:

```
####### Step 5 ################ -  Entity resolution for Event Kg and Event registry
```

In [66]:

```
def entity_kg_resolution(event_kg_mapped_keyword_list, sparql_query):
    keyowrd_list_with_ques_mark = []
    for each in event_kg_mapped_keyword_list:
        each = ''.join(('?',each))
        keyowrd_list_with_ques_mark.append(each)

    keyowrd_list_with_ques_mark.insert(0, "")

    keyword_list_with_space = " ".join(keyowrd_list_with_ques_mark)
    keyword_list_with_space_length = len(keyword_list_with_space)+1
    keyword_list_with_space = keyword_list_with_space.ljust(keyword_list_with_space_length)

    sparql_query_resolved = ("".join(re.findall(r"SELECT(.+?)WHERE" , sparql_query)))
    sparql_query_resolved = sparql_query.replace(sparql_query_resolved,keyword_list_with_space)
    return sparql_query_resolved
```

In [67]:

```
sparql_query_resolved_event_kg = (entity_kg_resolution(event_kg_mapped_keyword_list,sparql_query))
sparql_query_resolved_event_kg
```

Out[67]:

```
'PREFIX eventKG-s: <http://eventKG.l3s.uni-hannover.de/schema/> PREFIX eventKG-g: <http://eventKG.l3s.uni-hannover.de/graph/> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX so: <http://schema.org/> PREFIX sem: <http://semanticweb.cs.vu.nl/2009/11/sem/> PREFIX owl: <http://www.w3.org/2002/07/owl#> PREFIX dbr: <http://dbpedia.org/resource/>  SELECT ?location ?startdate WHERE {  ?event rdf:type sem:Event .  ?event owl:sameAs dbr:Battle_of_Trafalgar .  GRAPH eventKG-g:event_kg {?event sem:hasPlace ?loc} .  GRAPH eventKG-g:dbpedia_en { ?loc owl:sameAs ?location . } }'
```

In [ ]:

```
```

In [68]:

```
def entity_registry_resolution(event_registry_mapped_keyword_list, sparql_query):
    keyowrd_list_with_ques_mark = []
    for each in event_registry_mapped_keyword_list:
        each = ''.join(('?',each))
        keyowrd_list_with_ques_mark.append(each)

    keyowrd_list_with_ques_mark.insert(0, "")

    keyword_list_with_space = " ".join(keyowrd_list_with_ques_mark)
    keyword_list_with_space_length = len(keyword_list_with_space)+1
    keyword_list_with_space = keyword_list_with_space.ljust(keyword_list_with_space_length)

    sparql_query_resolved = ("".join(re.findall(r"SELECT(.+?)WHERE" , sparql_query)))
    sparql_query_resolved = sparql_query.replace(sparql_query_resolved,keyword_list_with_space)
    return sparql_query_resolved
```

In [69]:

```
sparql_query_resolved_event_registry = (entity_registry_resolution(event_registry_mapped_keyword_list,sparql_query))
sparql_query_resolved_event_registry
```

Out[69]:

```
'PREFIX eventKG-s: <http://eventKG.l3s.uni-hannover.de/schema/> PREFIX eventKG-g: <http://eventKG.l3s.uni-hannover.de/graph/> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX so: <http://schema.org/> PREFIX sem: <http://semanticweb.cs.vu.nl/2009/11/sem/> PREFIX owl: <http://www.w3.org/2002/07/owl#> PREFIX dbr: <http://dbpedia.org/resource/>  SELECT ?location ?eventDate WHERE {  ?event rdf:type sem:Event .  ?event owl:sameAs dbr:Battle_of_Trafalgar .  GRAPH eventKG-g:event_kg {?event sem:hasPlace ?loc} .  GRAPH eventKG-g:dbpedia_en { ?loc owl:sameAs ?location . } }'
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
####### Step 6 ################ -   Event resolution for event registry
```

In [70]:

```
def event_resolution_event_registry(sparql_query_resolved_event_registry):

    event = re.findall(r"dbr:([^ <]+)",sparql_query_resolved_event_registry)
    event = "".join(event)
    event = event.replace("_", " ")


    return event
```

In [71]:

```
event_resolved_name = event_resolution_event_registry(sparql_query_resolved_event_registry)
event_resolved_name
```

Out[71]:

```
'Battle of Trafalgar'
```

In [ ]:

```

```

In [ ]:

```
####### Step 7 ################ -   Event Kg output
```

In [76]:

```
def event_kg_output(sparql_query_resolved_event_kg):
    sparql.setQuery(sparql_query_resolved_event_kg)
    sparql.setReturnFormat(XML)
    results = sparql.query().convert()
    #print(results.toxml())
    return results.toxml()
```

In [77]:

```
event_kg_output = event_kg_output(sparql_query_resolved_event_kg)
event_kg_output
```

Out[77]:


```
'<?xml version="1.0" ?><sparql xmlns="http://www.w3.org/2005/sparql-results#" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/sw/DataAccess/rf1/result2.xsd">
\n <head>\n  <variable name="location"/>\n  <variable name="startdate"/>\n </head>\n <results distinct=
"false" ordered="true">\n  <result>\n   <binding name="location"><uri>http://dbpedia.org/resource/Atlan
tic_Ocean</uri></binding>\n  </result>\n  <result>\n   <binding name="location"><uri>http://dbpedia.org
/resource/First_French_Empire</uri></binding>\n  </result>\n  <result>\n   <binding name="location"><ur
i>http://dbpedia.org/resource/Strait_of_Gibraltar</uri></binding>\n  </result>\n  <result>\n   <binding
name="location"><uri>http://dbpedia.org/resource/Cape_Trafalgar</uri></binding>\n  </result>\n  <result
```

```
>\n   <binding name="location"><uri>http://dbpedia.org/resource/Gulf_of_Cádiz</uri></binding>\n  </resu
lt>\n </results>\n</sparql>'
```

In [ ]:

```
```

In [ ]:

```
####### Step 8 ################ -   Event registry output
```

In [88]:

```
def event_registry_output(event_resolved_name):
    iter = QueryEventsIter(conceptUri = er.getConceptUri(event_resolved_name))
    for art in iter.execQuery(er, sortBy = "rel"):
        #print(art)
        event_output = (json.dumps(art, indent=4))

        break

    event_output = json.loads(event_output)

    for i in event_output["concepts"]:

        if i["type"] == "loc":
            location = (i["label"]["eng"])

            break
    starttime = event_output["eventDate"]
    endtime = event_output["eventDate"]

    return {
        "location" : location,
        "eventDate" : starttime,
        "enddate" : endtime


    }
```

In [89]:

```
event_registry_output = event_registry_output(event_resolved_name)
event_registry_output
```

Out[89]:

```
{'location': 'London', 'eventDate': '2019-09-11', 'enddate': '2019-09-11'}
```

In [90]:

```
print("The ouput from event registry is as follows : ")
for ent in event_registry_mapped_keyword_list:
    if ent in event_registry_output:

        print(ent ,":" ,event_registry_output[ent])
```

```
The ouput from event registry is as follows :
location : London
```

```
location : London
eventDate : 2019-09-11
```

In [ ]:

```
```