Written Assignment 4: Create a Wikipedia Page

Mahathi Vempati, 20161003

November 14, 2019

1 Introduction

Wikidata is a collection of data organised in the form of a knowledge graph. Every relation is expressed in the form of a 3-tuple (wikidata-item, relation-to, another wikidata-item). The page of every wikidata item has the relations that are associated with it. Every relation is given an id starting with P, and every wikidata item is given an id starting with Q. For example, Douglas Adams is Q42. Each wiki data item is available in as many languages as possible.

The task at hand is to make a readable Wikipedia page, in Hindi, or any other Indian language from Wikidata using querying languages on Wikidata such as SPARQL.

How would one go about this?

2 Process

First, we pick a topic. Assume in this case, we want to pick the mathematician Ramanujan. First, we attempt to get all level 1 data about him. So, we query for all relations that are of the form (Ramanujan, any relation, any Wiki data).

```
import requests

url = 'https://query.wikidata.org/sparql'
query = """
PREFIX entity: <http://www.wikidata.org/entity/>
#partial results

SELECT ?propUrl ?propLabel ?valUrl ?valLabel ?picture
WHERE
{
hint:Query hint:optimizer 'None' .
{ BIND(entity:Q83163 AS ?valUrl) .
BIND("N/A" AS ?propUrl ) .
```

```
BIND("identity"@en AS ?propLabel ) .
}
UNION
{ entity:Q83163 ?propUrl ?valUrl .
?property ?ref ?propUrl .
?property rdf:type wikibase:Property .
?property rdfs:label ?propLabel
}

   ?valUrl rdfs:label ?valLabel
FILTER (LANG(?valLabel) = 'hi') .
OPTIONAL{ ?valUrl wdt:P18 ?picture .}
FILTER (lang(?propLabel) = 'hi' )
}
ORDER BY ?propUrl ?valUrl

"""
```

This outputs a few relations which I format and write to a file.

```
    r = requests.get(url, params = {'format': 'json', 'query': query})
data = r.json()
with open("ram.json", "w") as f:
    f.write(str(data))

info = {}
for item in data['results']['bindings']:
    info[item['propLabel']['value']] = item['valLabel']['value']

with open("ramanujan_page", "w") as f:
    for key in info.keys():
        f.write("Ramanujan ka"+key+" "+info[key]+" hai|\n")
print(info)
```

3 Going further

Now, we would like to add another layer of depth. To do this, we iterate through each item returned in the previous query, and query it again recursively, so that more information about each item is displayed.

```
l2_items = set()
```

```
for item in data['results']['bindings']:
    data_item = item['valUrl']['value'].split('/')[-1]
    val = item['valLabel']['value']
    l2_items.add((data_item, val))

for di in l2_items:
    query = """
PREFIX entity: <http://www.wikidata.org/entity/>
#partial results

SELECT ?propUrl ?propLabel ?valUrl ?valLabel ?picture
WHERE
{
hint:Query hint:optimizer 'None' .
{ BIND(entity:"""+di[0]+""" AS ?valUrl) .
BIND("N/A" AS ?propUrl ) .
BIND("identity"@en AS ?propLabel ) .
}
UNION
{ entity:"""+di[0]+""" ?propUrl ?valUrl .
?property ?ref ?propUrl .
?property rdf:type wikibase:Property .
?property rdfs:label ?propLabel
}

   ?valUrl rdfs:label ?valLabel
FILTER (LANG(?valLabel) = 'hi') .
OPTIONAL{ ?valUrl wdt:P18 ?picture .}
FILTER (lang(?propLabel) = 'hi' )
}
ORDER BY ?propUrl ?valUrl

"""
    r = requests.get(url, params = {'format': 'json', 'query': query})
    data = r.json()
    print(data)
    print("-"*50)

    info = {}
    for item in data['results']['bindings']:
        info[item['propLabel']['value']] = item['valLabel']['value']
```

```
with open("ramanujan_page", "a") as f:
    for key in info.keys():
        f.write(di[1]+ "  "+key+" "+info[key]+" |\n")
    f.write("\n")
```

The resultant page now has two layers of data.

4 Possible Improvements

- Transliterate any direct English written words that occur. (Done manually this time)
- Use language models to automatically construct sentences instead of the standard used here. An example is shown here. Since these are more commonly available in English, first completing the task in English, and then a direct translation can be explored.
- Writing several queries for the initial first topic can be explored.