

Introduction to Semantic Systems
2019W
188.399

Group 24
Final project: Healthcare App

Armin Gander - 11848230
Hubert Baginski - 01551118
Peter Bognar - 11846044
Christine Ning - 01204350

14. Februar 2020

Contents

1	Contribution	3
2	Application Description	3
2.1	Competency questions	3
3	Datasets	3
3.1	Dataset acquisition	3
3.1.1	Doctors data	3
3.1.2	Disease data	4
3.1.3	Patient data	4
3.1.4	Drug data	5
3.1.5	Street name data	5
4	Ontology	5
5	Creating a Knowledge Graph	7
5.1	Technology stack	8
6	CONSTRUCT queries to enrich knowledge of KG	8
7	Knowledge Graph Queries	9
7.1	Query 1	9
7.2	Query 2	9
7.3	Query 3	10
7.4	Query 4	11
7.5	Query 5	12
7.6	Query 6	13
7.7	Query 7	13
7.8	Query 8	14
7.9	Query 9	15
7.10	Query 10	15
7.11	Query 11	16
7.12	Query 12	16
8	Web Application	17
9	Conclusion	18

1 Contribution

Except for collecting the datasets and writing the queries, we worked on the other tasks as a team. As for the dataset, collecting the doctors data set was done by Christine, the disease data set by Peter, the patient and street name data by Armin and the drug data by Hubert.

2 Application Description

Our goal is to create a medical app which utilises the strengths of the semantic web to create a "single source of truth" for patients. Therefore, we combine various data sources regarding medical information, geographical information, personal information and information about doctors. To limit the scope of the project, we focus on Vienna. The aim of this app is to function as a central source of information for people interesting in general and person-specific details about diseases and access to healthcare services nearby.

2.1 Competency questions

The following list enumerates some competency questions which we initially planned to answer with our final knowledge graph:

- Which doctors can treat my illness XXX?
- Which doctor is the best (best rating) and can treat my illness?
- Where is the doctors office/hospital?
- Which alternative medicines to some medicine XXX are there?
- Which disease can I have given a set of symptoms?
- How many doctors are there in my district?
- How many doctors are there in my district which can treat my illness?

3 Datasets

Dataset	Source
Doctors data (Austria)	www.docfinder.at
Disease data	people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html
Patient data	www.kaggle.com/karimnahas/medicaldata
Drug data	www.drugbank.ca
Street names (Vienna)	https://www.wien.gv.at/statistik/verkehr-wohnen/strassen/

3.1 Dataset acquisition

3.1.1 Doctors data

For the doctors data, scrapy - an open source framework written in Python for extracting data from websites - was used (see folder /Source/data-preprocessing/semantic). We chose the website

www.docfinder.at to extract relevant data of doctors in Austria. Since one would get at most 100 search findings shown on 10 pages each search, we also collected medical speciality and the combination of zipcode and city name for an extensive collection of doctors in Austria. The medical speciality available on the website was collected manually. We distinguished common speciality with more than 100 doctors with this speciality in Austria and those with less. For the cities, a list of zipcodes in Austria was downloaded from the government website (<https://www.data.gv.at/katalog/dataset/f76ed887-00d6-450f-a158-9f8b1cbbefbf>). Using the zipcodes the combination of zipcodes and city name used on the website was scraped (e.g. <https://www.docfinder.at/search/autosuggest/where?term=1010> with 1010 being the zipcode). The umlauts in German were replaced and space was replaced with hyphen (e.g. scraped term “3345 Göstling an der Ybbs“ was replaced with “3345-Goestling-an-der-Ybbs“). With the combination of common medical speciality and place (zipcode + city name) as well as less common medical speciality without place the URLs for scraping were created (e.g. www.docfinder.at/suche/1010-Wien/praktischer-arzt or www.docfinder.at/suche/nuklearmediziner?whereType=country). The links to the doctors on each search page were used for the next request to collect data on the website of a certain doctor. While scraping we encountered an error since no items were returned for cities with “st.“ in name. Then we noticed that the name should be replaced with “st“ (e.g. 4300-st.-valentin to 4300-st-valentin). We extracted all cities with “St.“ in name and run the spider for only these cities with the corrected terms again. After deleting duplicates, we got 20586 doctors with the following information as a csv-file: name, phone number, address, amount of rating, rating, further education, opening hours on each weekday, medical speciality, certificate and medical insurance.

3.1.2 Disease data

The disease data was scraped from the following website: people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html. As we can see the website contains a main table which gives us the information, which symptoms can be examined in case of different diseases. The table has 134 diseases and its corresponding symptoms. We used the BeautifulSoup HTML parser library to fetch the information from the HTML file. The script `disease_parser.py` which we wrote and used is in the `/Source/data-preprocessing/` folder. The output of the script is in CSV format, where the first column is the disease and the following ones are the symptoms of the given disease:

```
disease1, symptom1, symptom2, ... symptomN
disease2, symptom1, symptom2, ... symptomN ...
```

3.1.3 Patient data

For the patient data we downloaded the open data from kaggle as a csv-file containing 2000 patients. Not surprisingly, the data were pseudoanonymized with the following information: name, gender, date of birth, zipcode, employment status, education, marital status, children, ancestry, average commute, daily internet use, available vehicles, military service and disease. We replaced the generated zipcodes of the patient with random real zipcodes of Vienna for patient living in Austria to link to the doctor data. Furthermore, we generated active substance the patient is allergic to to link the data to the drug data.

Since in our patient data, we have four diseases (endometriosis, multiple sclerosis, skin cancer and heart disease) which are not listed in the scraped disease data, we added these diseases

as well as its symptoms (through web search) to our disease data for a better demonstration of the knowledge graph queries. Some of the disease name in the disease data did not match with the patient data. Thus, we changed the disease in the patient data to link it to the disease data (e.g. “skin cancer“ to “malignant neoplasm of skin“ or “hypertension“ to “hypertensive disease“) in order to get the symptoms corresponding to the disease the patient (we assume) to have. In order to link the disease with the medical speciality of the doctors a extra table is created for those thirteen distinct diseases occurring in the patient data with disease name and medical specialities (e.g. disease “schizophrenia“ with the corresponding medical speciality in German “Neurologin, Neurologe, Psychiater, Psychiaterin“).

3.1.4 Drug data

For the drug data we downloaded the XML file from the open source drugbank data. However, we encountered problems with reading the 1GB+ XML file since there was an error in some lines in the data. We later found a R package (dbdataset) with the already parsed datasets of drugbank data (daiyanahan.github.io/dbdataset/articles/dbdataset.html). Since it contained 75 dataframes with different information like reference of carriers, transporters and enzymes, interaction with food and drugs, manufacturer and so on, we selected some of the dataframes and columns like description of drug, drug substance, drug manufacturer, etc. which in our opinion are relevant to answer our competency questions and selected a subset of it due to the big size. Our final decision was to work with a subset of the data of 1000 drugs.

3.1.5 Street name data

The street name of Vienna are saved as a csv file downloaded from open data Austria with all in all 12226 street names and its corresponding district. We originally planned to query the place not just by zipcode but also by its distance. However, it was hard to extract all the longitude and latitude information of the addresses. Thus, we discarded the original plan and stayed with the zipcode.

4 Ontology

When modelling our ontology we oriented ourselves on *Ontology Development 101: A Guide to Creating Your First Ontology* by Natalya F. Noy and Deborah L. McGuinness.

We started by determining the scope and domain of our ontology. Healthcare is our domain. Determining the scope was more difficult. Initially we wanted to include data about all medical professionals in Austria, but due to difficulties in data acquisition we decided to limit it on Vienna. Furthermore, we were not sure about how detailed our ontology should be and which information should be included as entities and which information is not relevant for us. We ‘solved’ this problem by reworking our ontology iteratively.

Next we enumerated the terms which are needed in our ontology. We also looked for already existing ontologies to reuse. This has many advantages, saving effort and increasing the semantic interoperability between systems amongst others. In the end, we reused two of the most popular ontologies:

1. Friend of a friend: <http://xmlns.com/foaf/spec/20140114.html>
2. DBpedia: <http://dbpedia.org/ontology/>

We continued our process of modelling our ontology by defining the classes and the class hierarchy. For that, we used a combination of a top down and bottom up approach. Due to the iterative process of creating our ontology we modified our class hierarchy several times which always led to the rearrangement of classes and subclasses. In the end, our class hierarchy looked as shown in Fig. 1.

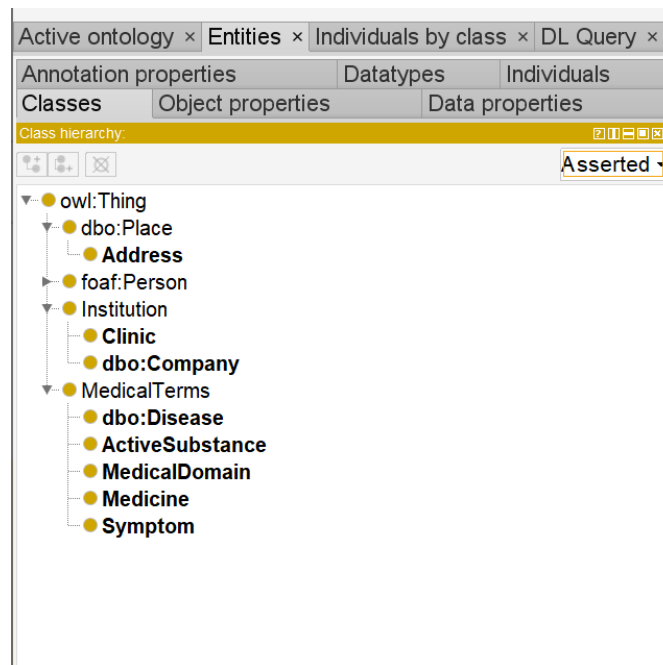


Fig. 1: Class hierarchy of our ontology

The next step consisted of defining the properties of classes. For that, we used the software *Protégé* and pen and paper. During this step we also started thinking about data properties and property restrictions. Lastly, we inserted the instances and checked of anomalies using the reasoner inbuilt in *Protégé*.

Our final ontology looks as shown in Fig. 2.

It can be clearly seen that the ontology can be separated into four clusters. The classes on the right (Place, Address and its properties) include all the geographical information needed in our knowledge graph. In the center top of our ontology in figure 2 we can see a collection of medical terms to describe illnesses and the connected terminology. Classes corresponding to persons (doctors and patients) belonging to our knowledge graph are clustered on the bottom left. The final cluster of classes corresponds to institutions which connect doctors with the addresses of their offices.

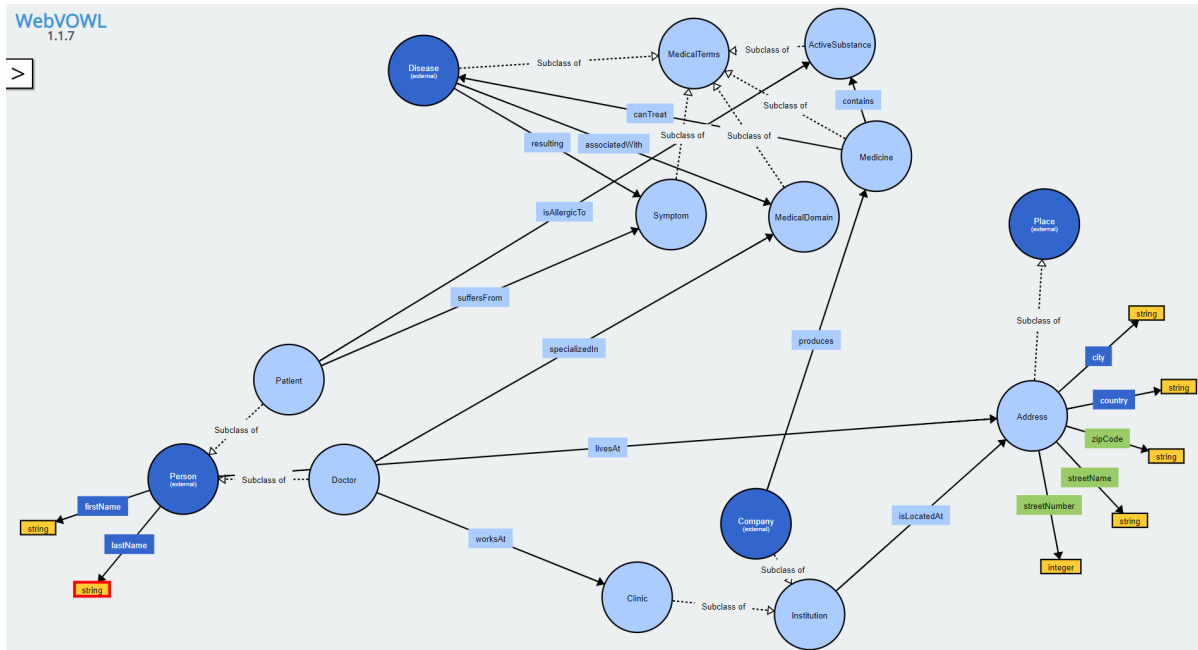


Fig. 2: Healthcare ontology

5 Creating a Knowledge Graph

We worked together on the processing and RDF triple creation, but every group member installed OpenRefine locally and was responsible for one dataset: Armin - patient and street names, Christine - doctors, Hubert - drugs, Peter - diseases.

This section focuses on our approach of handling multiple different datasets with respect to their size, data format and data quality. We experimented with the different tools which were recommended to us, but ended up using OpenRefine¹. Once we had installed the RDF extension we were able to create the first RDF files.

Whilst working with multiple datasets the triples which we had to create in order to inter-link the data correctly, i.e. such that we would be able to answer our research questions, turned out to be the most difficult, and most important step in this task. Through exhaustive experimentation (and a lot of failure) we developed the following methodology of importing, cleaning, and exporting the desired triples into RDF: First, we loaded the data (mostly CSVs and one XML file, can be found in /Data/DrugBank, /Data/geo, /Data/patient, and in /Data) into OpenRefine, which allowed us to inspect the datasets more closely. Second, we pre-processed the data such that only information which we had defined in our ontology remained. Subsequently, we also performed minor text pre-processing (removal and replacement of unwanted characters, lower-casing, splitting and joining of columns) which would allow us to write the SPARQL queries in a clearly defined format. Third, we defined the RDF skeleton for each dataset according to our ontology. Consequently, we created four RDF files for **Patient**, **Doctor**, **Medicine**, and **Disease**. The final RDF files can be found in the following path: /Data/RDF. All of the pre-processing was done via the UI of OpenRefine, thus no code is provided. The results, RDF/XML

¹<https://openrefine.org/>

files can be found in /Data/RDF.

5.1 Technology stack

We used OpenRefine and the RDFRefine extension. We imported the CSV and XML files into OpenRefine, performed data analysis, exploration and pre-processing. Last, we defined the optimal RDF skeleton structure and exported the triples as RDF/XML files.

6 CONSTRUCT queries to enrich knowledge of KG

We worked together on the KG creation and query development, but every group member installed GraphDB locally and was responsible for three queries, work distribution explained in detail in **section 6**.

We chose GraphDB ² as our knowledge graph. First, (after installing the software on our local machines) we created a repository in GraphDB and imported the four RDF/XML files (patients, doctors, medicine, diseases) into the default graph.

Since we had spent a lot of time in the creation of the RDF files in Task 4, we had very little deviation from our KG and our ontology. Thus, we decided that the most sensible step would be to create inverse links between e.g. a patient with symptoms and symptoms which occurred in patients. We ran the following queries in order to enrich our knowledge graph:

```
PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT {
    ?Symptom :isPresent ?Patient.
} |
WHERE {
    ?Patient :suffersFrom ?Symptom.
}

PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT {
    ?Disease :canBeTreatedWith ?Medicine.
}
WHERE {
    ?Medicine :canTreat ?Disease.
}
```

²<http://graphdb.ontotext.com/>

7 Knowledge Graph Queries

We developed the following queries in order to i) answer our defined research questions and ii) to answer additional interesting questions which we could now answer with our enriched KG. We worked closely together developing the queries and it was a collaborative effort. However, since it is required we have split the developed queries as follows:

- Armin: Query 1, Query 6, Query 7
- Christine: Query 2, Query 3, Query 9
- Hubert: Query 4, Query 11, Query 12
- Peter: Query 5, Query 8, Query 10

7.1 Query 1

The first query is our attempt to beat Google. We want to determine which disease the patient has (or might have) based on the symptoms he has. Thus, we simulated the symptoms via a filter condition and tried to answer:

Which disease can a patient have given a set of symptoms?

```
PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
    ?disease foaf:name ?name;
             :resulting ?symptom.
    FILTER (CONTAINS(?symptom, "pain chest") && CONTAINS(?symptom, "dizziness"))
}
```

	name	
1	hypertensive disease	
2	hyperlipidemia	

7.2 Query 2

This query focused on the patients which we have in our KG. They have some demographic information such as name, age, etc. and a set of symptoms. We now use the link between symptoms and diseases and return the people with a list of disease which contains a specific symptom:

Which patients have the symptom *"nightmare"* and what are the possible diseases linked with nightmares?

```

PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?firstname ?lastname ?symptom (GROUP_CONCAT(?name;separator=" | ") as ?possible_diseases)
WHERE {
    ?Patient foaf:firstName ?firstname;
             foaf:lastName ?lastname;
             :suffersFrom ?symptom.
    ?disease foaf:name ?name;
             :resulting ?symptom1.
    FILTER (CONTAINS(?symptom, "nightmare") && CONTAINS(?symptom1, "nightmare"))
}

GROUP BY ?firstname ?lastname ?symptom

```

	firstname	lastname	symptom	possible_diseases
	↕	↕	↓ _Z ^A	↕
1	Sofia	Wise	nightmare	depression mental hypothyroidism anxiety state psychotic disorder Alzheimer's disease personality dis order
2	Adam	Strickland	nightmare	depression mental hypothyroidism anxiety state psychotic disorder Alzheimer's disease personality dis order
3	Emma	Cook	nightmare	depression mental hypothyroidism anxiety state psychotic disorder Alzheimer's disease personality dis order
4	Violet	Hernande z	nightmare	depression mental hypothyroidism anxiety state psychotic disorder Alzheimer's disease personality dis order

7.3 Query 3

We wanted to focus on the following case: The user of our application has already entered his symptoms and knows which disease he has. In this case he has skin cancer and must visit a dermatologist. Our dataset was in German, and the scope was reduced to doctors in Vienna,

thus, we filtered on "*Haut*", which translates to "*skin*". This was done because we have Hautarzt and Hautärztin. We thus answer our competency question (rather primitively):

Which doctor can treat my illness?

```
PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?lastname ?address ?phone (GROUP_CONCAT(?specialty;separator=" | ") as ?specialties)
WHERE {
    ?Doctor foaf:lastName ?lastname;
           foaf:phone ?phone;
           :worksAt ?address;
           :specializedIn ?specialty.
    FILTER (CONTAINS(?specialty, "Haut"))
}
GROUP BY ?lastname ?address ?phone
```

Filter query results		Showing results from 1 to 624 of 624. Query took 0.7s, moments ago.		
	lastname ↕	address ↕	phone ↕	specialties ↕
1	Dr. Zsuzsanna Palfi	Vorgartenstraße 206C, 1020 Wien	01/890 45 20	Hautärztin Hautärztin,Wahlärztin
2	Dr. Bernd Gmeinhardt	Rembrandtstraße 12/4, 1020 Wien	01/330 45 05	Hautarzt
3	Dr. Michael Younan	Vorgartenstraße 206C, 1020 Wien	01/890 45 20	Hautarzt Hautarzt,Wahlarzt
4	Dr. Sigrid Psailer	Fanny-Mintz-Gasse 3/Top 10, 1020 Wien	01/2123005	Hautärztin Hautärztin,Wahlärztin
5	Dr. Josefine Herta Klade	Praterstraße 56, 1020 Wien	01/212 68 58	Hautärztin
6	Dr. Helmut Leonhartsberger	Praterstraße 10/5, 1020 Wien	01/214 13 86	Hautarzt
7	Dr. Anna Gappmayer	Große Pfarrgasse 23 / 1, 1020 Wien	0660/325444 8	Hautärztin Hautärztin,Wahlärztin

This question was also answered more elaborately, exploiting the linked data structure of our KG, in **Query 9** and **Query 12**.

7.4 Query 4

The next aspect we wanted to research was which drugs can be used as treatment for my condition. We had a gigantic dataset from Canada's drug bank, which proved to be too big for both the reconciliation in OpenRefine and for the subsequent queries in GraphDB. Thus, we decided to work with a subset of 1000 drugs. Furthermore, there was no field which explicitly mentioned which disease is treated by the drug, but we were able to solve this by using the textual description of the drug/treatment. Consequently, we use partial string matching in our FILTER condition and answer the following question;

Which drugs can I use to treat my illness?

```

PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?drug ?treat
WHERE {
    ?Medicine foaf:name ?drug;
              :canTreat ?treat.
    FILTER (CONTAINS(?treat, "treatment of skin"))
}

```

	drug	treat
1	Gramicidin D	For treatment of skin lesions, surface wounds and eye infections.
2	deep vein thrombosis	For treatment of skin lesions, surface wounds and eye infections.
3	Becaplermin	For topical treatment of skin ulcers (from diabetes)
4	fibroid tumor	For topical treatment of skin ulcers (from diabetes)
5	Bexarotene	Used orally for the treatment of skin manifestations of cutaneous T-cell lymphoma (CTCL) in patients who are refractory to at least one prior systemic therapy. Also used topically for the treatment of skin lesions in early (stage IA and IB) CTCL in patients who experience refractory or persistent disease with the use of other therapies or are intolerant of other therapies.
6	Ciprofloxacin	Ciprofloxacin is only indicated in infections caused by susceptible bacteria.[L6469,L6472,L6475,L6478,L6481,L6484,L6487,L6490,L6493]

7.5 Query 5

This query focuses on the doctors, more precisely their offices or clinics where they are working. We wanted to help the user answer the question:

Where is the doctors office / hospital?

```

PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT (?lastname as ?full_name) ?address ?phone
      (GROUP_CONCAT(DISTINCT?specialty;separator=" | ") as ?specialties)
WHERE {
    ?Doctor foaf:lastName ?lastname;
            foaf:phone ?phone;
            :worksAt ?address;
            :specializedIn ?specialty.
}
GROUP BY ?lastname ?address ?phone
LIMIT 25

```

	full_name	address	phone	specialties
1	Lic.Med. Carolina Martin Mestre	Fanny-Mintz-Gasse 3, 1020 wien	01/2123005	Anästhesistin,Wahlärztin
2	Dr. Wilfrid Junker	Vorgartenstraße 158/1/1, 1020 Wien	01/728 68 81	Lungenfacharzt
3	Priv.Do. Univ.Prof.DPU Dr. Dritan Turhani	Obere Donaustraße 97-99/4/3, 1020 Wien	0664/5308060	Kieferchirurg,Wahlarzt
4	Dr. Luka Girardi	Schoellerhofgasse 5, 1020 Wien	0676/ 91 35 664	Internist, Angiologe, Geriater,Wahlarzt
5	Dr. Zsuzsanna Palfi	Vorgartenstraße 206C, 1020 Wien	01/890 45 20	Hautärztin,Wahlärztin
6	OA Dr. Michael Vrba	Obere Donaustraße 43/1/23, 1020 Wien	0664/211 88 88	Allgemeinchirurg
7	Dr. Petros Skyllouriotis	Praterstraße 78/2/5, 1020 Wien	0664/454 30 12	Allgemeinchirurg, Herzchirurg, Gefäßchirurg,Wahlarzt
8	OA Dr. Peter Schwameis	Vorgartenstraße 206C, 1020 Wien	01/890 45 20	Allgemeinchirurg,Wahlarzt

7.6 Query 6

This query focuses on patient with certain symptoms searching for a suitable doctor. It will link the symptoms to diseases and their corresponding medical domains. Thus, we answer:

Which doctors in a certain region (zipcode) should a certain patient go to with her/his symptoms?

```

PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT (?pname as ?doctor) ?address ?phone ?domain
WHERE { ?Doctor      foaf:lastName ?lname;foaf:phone ?phone;foaf:status ?rating;
                  :worksAt ?address;
                  :specializedIn ?domain.
      ?Disease      foaf:name ?disease;
                  :resulting ?symptoms;
                  :associatedWith ?domain1.
      ?Patient      foaf:lastName ?pname; foaf:firstName ?pfname;
                  :suffersFrom ?psymptoms.
      FILTER ( ?pname="Perry" && ?pfname="Nevaeh" && CONTAINS(?domain1, ?domain)
              && CONTAINS(?symptoms,?psymptoms) && CONTAINS(?address, "1030 wien"))}
ORDER BY DESC(?rating)

```

	doctor	address	phone	domain
1	Dr. Rudolf Hölzel	boerhaavegasse 21, 1030 wien	0664/5279624	Urologe
2	OA Dr. Clemens Hammer	barichgasse 22/6b, 1030 wien	01/7185777	Urologe
3	OA Dr. Christian Türk	ziehrerplatz 7/7, 1030 wien	01/712 65 74	Urologe
4	Prim. Dr. Rudolf Hasun	juchgasse 24, 1030 wien	01/7144631	Urologe
5	Dr. Marija Kamidzoric-Grbovic	rennweg 58/ top 1+2, 1030 wien	01/7156316	Urologin
6	MR Dr. Gregor Hienert	untere viaduktgasse 10/6, 1030 wien	01/7123122	Urologe

7.7 Query 7

A specific patient has a set of symptoms and a set of allergies to the active substance in drugs.

Which alternative drugs can he take, which are used to treat his illness, and he is not allergic to?

```
PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?drug
WHERE { ?Patient    foaf:firstName ?fname;
                    foaf:lastName ?lname;
                    :suffersFrom ?symptoms;
                    :isAllergicTo ?activeSubstance.
    ?Medicine      foaf:name ?drug;
                    :contains ?activeSubstanceDrug;
                    :canTreat ?disease.
    ?Disease       foaf:name ?disease1;
                    :resulting ?symptoms1.
    FILTER ( ?lname="Nixon" && ?activeSubstanceDrug!=?activeSubstance &&
    CONTAINS(?disease,?disease1) && CONTAINS(?symptoms1,?symptoms) )}
```

	drug
1	Phosphatidyl serine
2	NADH
3	Tacrine
4	Galantamine
5	Donepezil
6	Rivastigmine

7.8 Query 8

Which phone number should a specific patient with a set of symptoms call (number of a doctor which can treat his disease)?

```
PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?docname ?phone
WHERE { ?Patient    foaf:firstName ?fname; foaf:lastName ?lname;
                    :suffersFrom ?symptoms.
    ?Doctor        foaf:lastName ?docname; foaf:phone ?phone;
                    :worksAt ?clinic; :specializedIn ?medDomain.
    ?Disease       foaf:name ?disease;
                    :resulting ?symptoms1;
                    :associatedWith ?domain1.
    FILTER (CONTAINS(?fname,"Amelia") && CONTAINS(?lname,"Nixon") &&
    CONTAINS(?domain1, ?medDomain) && CONTAINS(?symptoms1,?symptoms) ))
}
```

LIMIT 50

	docname	phone
1	Dr. Veronica Adler-Stemberger	01/214 19 31
2	Dr. Dan Seidler	01/728 01 17
3	OA Dr. Peter Weimann	01/216 35 85
4	Dr. Walter Zuchristian	01/2181909
5	Dr. Johannes Fleischer	01/216 72 63
6	ao. Univ.Prof. Dr. Wolfgang Mlekusch	0699/10622695
7	MR Dr. Franz Widhalm	01/7135664
8	Dr. Ivan Drmic	01/7134920
9	Dr. Katharina Riedl	01/3070 666
10	Dr. Silke Böcskör	0699/196 98 491

7.9 Query 9

Which doctor is the best (best rating) for treating my symptoms?

```

PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT (?lname as ?doctor) ?rating ?domain
WHERE { ?Doctor      foaf:lastName ?lname;foaf:phone ?phone;foaf:status ?rating;
                  :worksAt ?address;
                  :specializedIn ?domain.
      ?Disease      foaf:name ?drug;
                  :resulting ?symptoms;
                  :associatedWith ?domain1
      FILTER ( CONTAINS(?domain1, ?domain) && CONTAINS(?symptoms,"vomiting") &&
                CONTAINS(?symptoms,"nausea") && CONTAINS(?symptoms,"pain abdominal"))
    }
ORDER BY DESC(?rating) LIMIT 10

```

	doctor	rating	domain
1	OA Dr. Johannes Kies	5.0	Internist
2	Dr. Marianne Stopar	5.0	Internistin
3	Dr. Markus Sedlak	5.0	Internist
4	Dr. Wolfgang Ziegelmeyer	5.0	Internist
5	Dr. Klaus Bernhofer	5.0	Internist
6	Dr. Gerlinde Lindner	5.0	Internistin
7	Dr. Barbara Dietze	5.0	Internistin
8	Dr. Bernhard Graf	5.0	Internist
9	Dr. Katarina Lirk	5.0	Internistin
10	Dr. Thomas Laimböck	5.0	Internist

7.10 Query 10

How many doctors are there in my district which can treat my illness?

```

PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT (COUNT(?lname) AS ?number_of_doctors)
WHERE { ?Doctor      foaf:lastName ?lname; foaf:phone ?phone; foaf:status ?rating;
                  :worksAt ?address;
                  :specializedIn ?domain.
        ?Disease     foaf:name ?disease;
                  :resulting ?symptoms;
                  :associatedWith ?domain1
        FILTER ( CONTAINS(?domain1, ?domain) && CONTAINS(?symptoms,"vomiting") &&
                  CONTAINS(?symptoms,"nausea") && CONTAINS(?symptoms,"pain abdominal")
                  && CONTAINS(?address, "1220 wien"))
        }

```

Filter query results		Showing results from 1 to 1 of 1. Query took 2.7s, moments ago.	
		number_of_doctors	
1	"4"^^xsd:integer		

7.11 Query 11

How many doctors are in my district (based on ZIP code)?

```

PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT (COUNT(?lname) AS ?number_of_doctors)
WHERE { ?Doctor      foaf:lastName ?lname;
                  foaf:phone ?phone;
                  foaf:status ?rating;
                  :worksAt ?address;
                  :specializedIn ?domain.
        FILTER ( CONTAINS(?address, "1220 wien") )
        }

```

		number_of_doctors	
1	"276"^^xsd:integer		

7.12 Query 12

Which doctor (best rating) should I visit who works nearby (near my ZIP code) and can treat my disease (combination of my symptoms)?


```

PREFIX : <http://www.semanticweb.org/christine/ontologies/2020/0/mdb#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT (?lname as ?doctor) ?phone ?address ?rating ?domain ?disease
WHERE { ?Doctor      foaf:lastName ?lname; foaf:phone ?phone; foaf:status ?rating;
                    :worksAt ?address;
                    :specializedIn ?domain.
      ?Disease      foaf:name ?disease;
                    :resulting ?symptoms;
                    :associatedWith ?domain1
      FILTER ( CONTAINS(?domain1, ?domain) && CONTAINS(?symptoms,"vomiting") &&
              CONTAINS(?symptoms,"nausea") && CONTAINS(?symptoms,"pain abdominal")
              && CONTAINS(?address, "1030 wien"))
    }
ORDER BY DESC(?rating)

```

Filter query results		Showing results from 1 to 5 of 5. Query took 3.5s, minutes ago.				
	doctor	phone	address	rating	domain	disease
1	Dr. Katharina Riedl	01/3070 666	am heumarkt 3/11, 1030 wien	5.0	Internistin	gastritis
2	Dr. Silke Böcskör	0699/196 98 491	apostelgasse 13, 1030 wien	5.0	Internistin	gastritis
3	Dr. Ladislaus Undesser	01/7983522	mohsgasse 4/3, 1030 wien	4.8	Internist	gastritis
4	MR Dr. Franz Widhalm	01/7135664	ungargasse 45/7, 1030 wien	4.2	Internist	gastritis
5	Dr. Pave Zanic	01/7134920	landstraßer hauptstraße 147/8, 1030 wien	4.2	Internist	gastritis

8 Web Application

We have created a simple web application which connects to our KG, performs the query and returns the result in form of text, enabling easy access to users without any technical knowledge. The application is written in JavaScript with the NodeJS framework.

We created a user interface with Materialize which gives the user the possibility to enter her/his symptoms in the form a free text field. The user has to type the symptoms in the given text field, separated by commas. The application then builds up the proper query and sends it to the KG, fetches the results and display them in a table.


HEALTH MASTER

EMERGENCY

DOCTORS NEARBY

MY DISEASES

MY SYMPTOMS



HEALTH MASTER

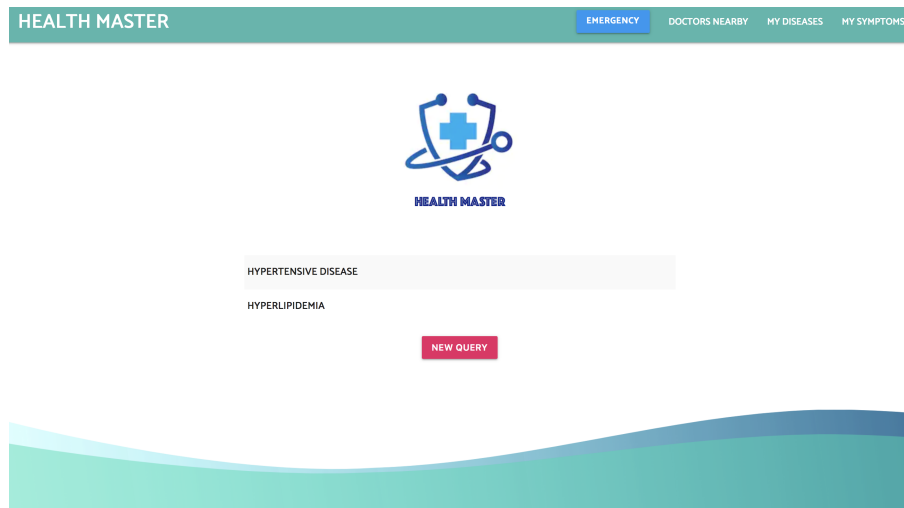
DO I HAVE A DISEASE?

PAIN CHEST, DIZZINESS

GO!

GraphDB must be running in order to connect to the KG and return the result of the query.

We also created a Python script³ which is using SPARQLWrapper⁴ to connect to the locally running repository. We then instantiate the two variables, **input1** and **input2** and run **Query 1**. The script runs the query with the user input and returns the results, the possible diseases which have the symptoms the user entered. We received the exact same results in both cases.



This is a very simple implementation, but the general idea remains the same. We provide the user with a simple text input, e.g. which ZIP code do you want to look for? Which symptoms do you have? Which drug are you allergic to? Subsequently, we would add the 12 queries we developed into the Python script, implement a check in the form of if statements which would select the correct query based on user input and run the query. We unanimously decided that if **Query 1** runs, all of them would run, thus, we decided not to implement them.

9 Conclusion

This project was the first concrete contact with linked data and semantic technologies in general for the members of our group. Initially, we did not understand the benefits of semantically linked data compared to a classic relational database. However, over the duration of the project we have learned to appreciate the semantic technologies, starting from easily interlinking data using OpenRefine's reconciliation services, to the easy and user-friendly data pre-processing and triple generation. Furthermore, once the RDF files were loaded into GraphDB the intuitive SPARQL syntax made the process of answering our competency questions simpler than working with a relational database (the fact that we don't have to care about joining multiple tables, join conditions and selections). Additionally, one can always extent the KG and add additional entities (data) to the KG by simply pre-processing the dataset and defining the triples according to the ontology.

However, we did not re-use any existing ontologies, triples or other resources. We decided, that for our first semantic project, that we should do everything, from scratch, ourselves. This provided a great learning experience, but in future projects we will **first** consider re-use of existing

³/Source/query1_webapp_userinput.py

⁴<https://rdflib.github.io/sparqlwrapper/>

ontologies and KG's, **then** we will try to start with a small ontology and work bottom-up, adding additional elements, classes and relations to the ontology when needed.

For some of us, we scraped the data for the first time, which was also something that we learned through the project. One must be a little creative in order to collect the data one is interested in, finding common features of the websites, detecting errors while scraping and e.g. circumvent the restricted pages shown on the websites. We also learned that we should think of how we should save the scraped data beforehand to make the pre-processing procedure easier later on.

Working in a team, we also learned that it is important that all of the team members are working with the same version and the actual data since otherwise conflicting results may occur.

Subsequently, we also learned (sadly the hard way), that even though one has access to different good datasets, interlinking them plays a crucial role and takes a long time also including simulating data to link the different data. Thus, our group spent the most time for the pre-processing and triple generation. However, since we spent a lot of time and effort there, the KG creation was as simple as importing all RDF/XML files into GraphDB and running the SPARQL queries.

Summarising, we were introduced to the semantic web, linked data and semantic systems through this course (and project). Initially, we had difficulties getting started since we could not clearly understand what was asked and what we ought to do exactly. However, the lectures provided great guidance (even though we worked ahead of the recommended time schedule for the project and got stuck a couple of times):) and made things clear. We all enjoyed working on this project and will, if the possibility arises, incorporate semantic technologies into future projects!