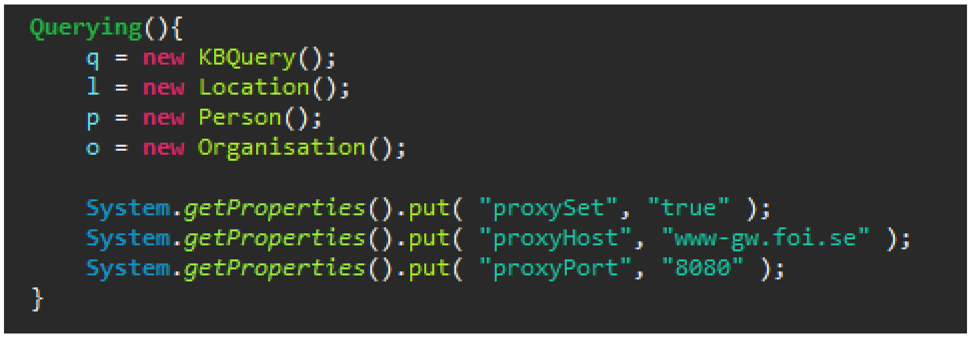
Documentation

Content

Installation and setup

1. Create a local Javaproject with the files from https://github.com/maaarrraaan/MartinErik . Use the .java files in the first folder, ignore the webapp and website folders.
2. Add the following internet setup to be able to access the online end point of DBpedia. This should be done in the constructor of the Querying class.

```
System.getProperties().put( "proxySet", "true" );
System.getProperties().put( "proxyHost", "www-gw.foi.se"
System.getProperties().put( "proxyPort", "8080"
```

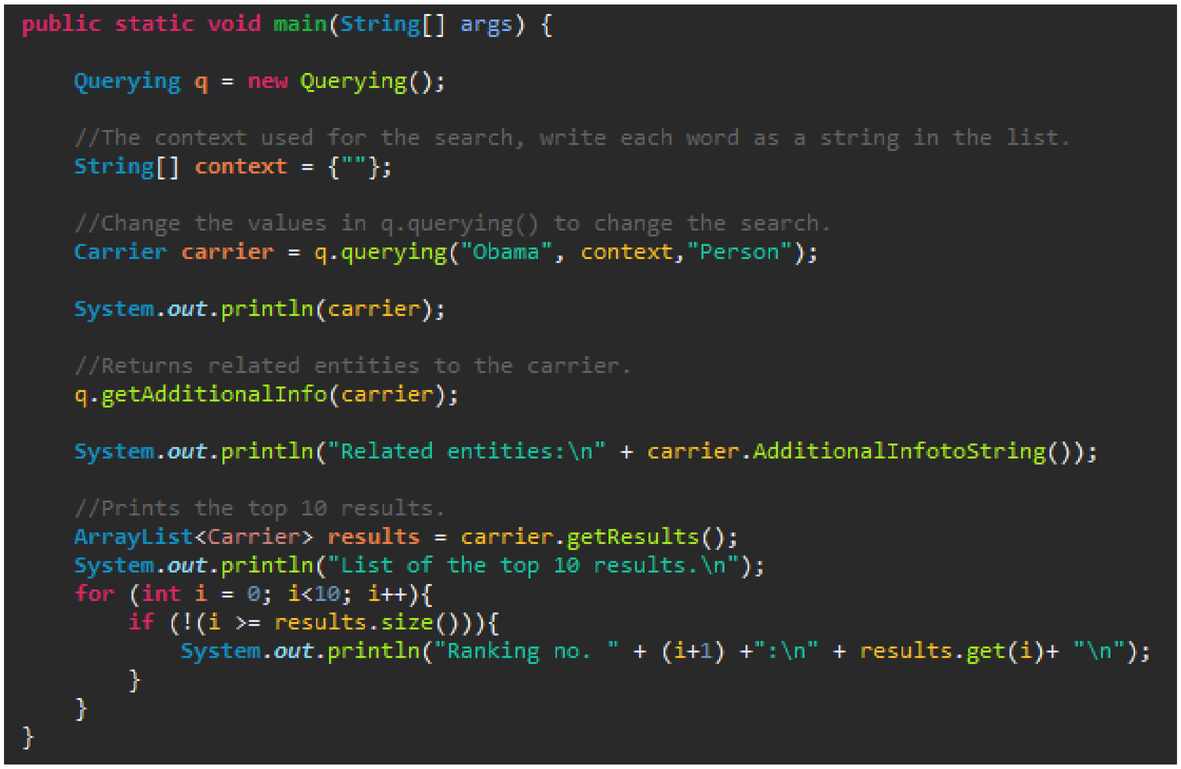


```
Querying(){
    q = new KBQuery();
    l = new Location();
    p = new Person();
    o = new Organisation();

    System.getProperties().put( "proxySet", "true" );
    System.getProperties().put( "proxyHost", "www-gw.foi.se" );
    System.getProperties().put( "proxyPort", "8080" );
}
```

3. Download and set up the external library Jena. It can be found here: https://jena.apache.org/

Using the interface

The picture below shows a simple implementation of the interface that simply prints the information to the consol. Yet this shows most of the different uses. Below the picture follows explanations of some of the functionality.



```java
public static void main(String[] args) {

    Querying q = new Querying();

    //The context used for the search, write each word as a string in the list.
    String[] context = {""};

    //Change the values in q.querying() to change the search.
    Carrier carrier = q.querying("Obama", context,"Person");

    System.out.println(carrier);

    //Returns related entities to the carrier.
    q.getAdditionalInfo(carrier);

    System.out.println("Related entities:\n" + carrier.AdditionalInfotoString());

    //Prints the top 10 results.
    ArrayList<Carrier> results = carrier.getResults();
    System.out.println("List of the top 10 results.\n");
    for (int i = 0; i<10; i++){
        if (!(i >= results.size())){
            System.out.println("Ranking no. " + (i+1) +":\n" + results.get(i)+ "\n");
        }
    }
}
```

Return best result

To return the best result follow the steps below. Before returning the answer the system will check the answer against the uncertainty limit. This is the recommended way to use the system as a part of a bigger system.

1. Create an object of the Querying class.
2. Call the objects function querying with the search term, the context and the type as input. The search term should be of the type organisation, person or location. The context is a list of strings that is relevant to the search term. Including to many ordinary word in the context risks less precision.
3. The system either returns an empty or filled carrier. If the carrier is empty it's either because the system is uncertain of the answer or because it couldn't find anything. To get the information see the section about how to interact with the carriers. A simple toString method is implemented for the carrier and can be used.

Return list of best results

In some cases it might be more intressting to get a list of possible answers. For instance in the case where the uncertainty limit is triggered or to present of list of other possibilities to a user.

1. Follow the instructions under the section "Return best result".
2. After the last step call the returned carrier function getResults(). This will return a sorted list of carriers.

Complementary searches

The information returned from a normal search can be complemented by an additional search that focuses on other related entities.

1. Get a carrier either by
    a. Follow the instructions under the section "Return best result".
    b. From the list of the best results.
    c. Some other way.
2. Call the getAdditionalInfo() in the querying object (if the carrier was accquiered through step 1 b or 1 c there might be a need to create a querying object from the Querying class). This function takes the carrier selected as input. The additional information is stored in the carrier.
3. Access the additional information through returnAdditionalInfo(). This function returns a map containing a key and the value connected to the key. Tip: A good way to loop through the map is map function keyset() which returns all keys in the map.

Running manually

A class running_query_manually is included as a simple way to test the system when developing. Change the parameters and get the answers printed straight in the consol.

Testing with your own test file

To test the system a testing function has been developed, Testing1. Testing1 is based on an external textfile. To conduct other tests the easiest way is to create a new file. Each row should be on the following format:

Search term%%Type%%Context separated by ';' (for no context '-')%%Correct answer

In the current version there are five different types for testing implemented. Each type has a different kind of answer. Which types there are and their kind of answer is presented in the table below.

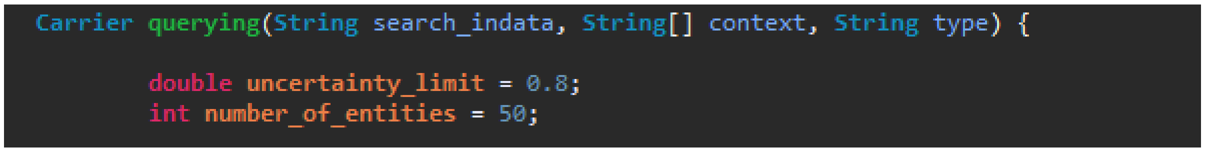
Type	**Kind of answer**
Person	Name
Person_more	Birth Date
Organisation	Name
Location_city	Country
Location_country	Total Population

The test row *Morowitz%%Person%%Biophysicist;Enigmas%%Harold J. Morowitz* illustrates an example of type person. Notice that the correct answer in this case is the name of the searched entity.

Change settings

Number of entities returned by topEntities

To change the number of entities returned by topEntities change the variable *number_of_entities* in the start of the class KBQuery. Default setting is 50. Increasing the number gives a might give a better coverage but at the cost of the speed of the program.



```
Carrier querying(String search_indata, String[] context, String type) {

        double uncertainty_limit = 0.8;
        int number_of_entities = 50;
```

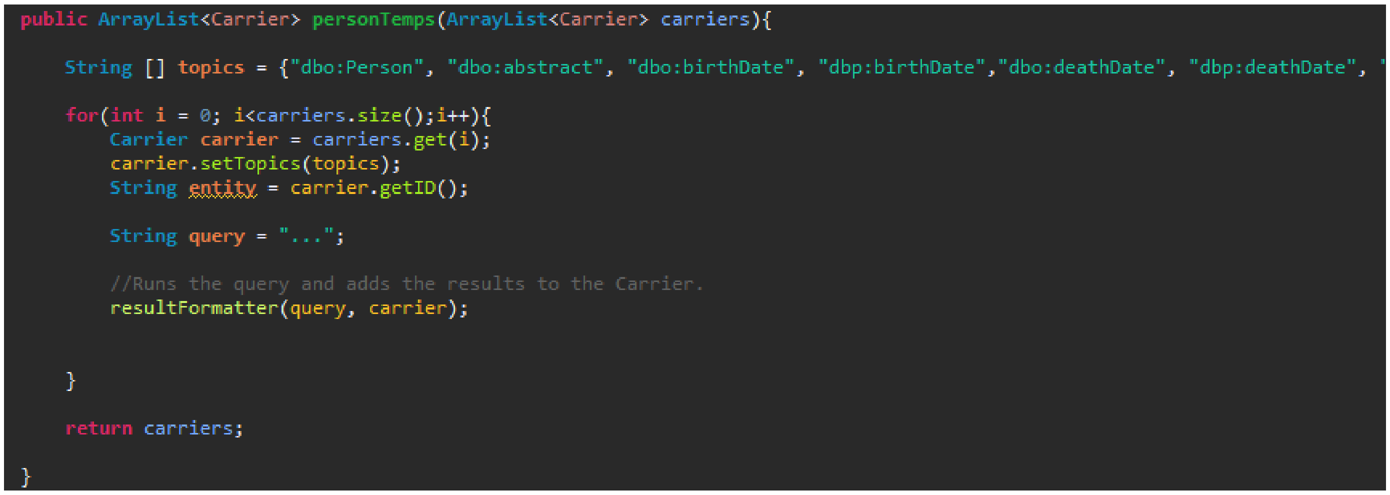
Uncertainty limit

To change the uncertainty limit change the variable *uncertainty_limit* in the start of the class KBQuery. Default value is 0.8. The value should be between 0 and 1. Setting the value at 0 leads to all questions being answered, no check of certainty is done. Setting the value at 1 returns only answers where the interface is completely certain it is right. This is not a guarantee that the answer will be right as the interface answers wrong in some cases.

Adding/Changing Templates

Adding additional templates involves two main steps: writing the new SPARQL template and adding the type. First of all creating the SPARQL template is often easier to do in some other environment than Java source. For the development of this system http://dbpedia.org/sparql have been used for this purpose.

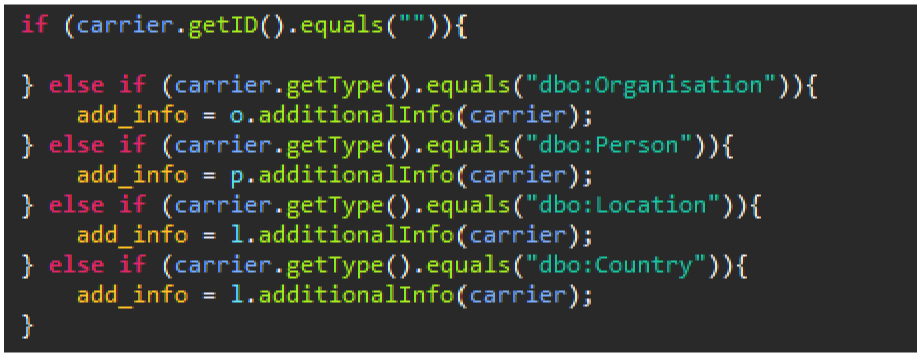
1. Create a new function in either any of the existing classes Person, Organisation or Location. The function will return a list of carriers and takes a list of carriers as input.



```
public ArrayList<Carrier> personTemps(ArrayList<Carrier> carriers){

    String [] topics = {"dbo:Person", "dbo:abstract", "dbo:birthDate", "dbp:birthDate","dbo:deathDate", "dbp:deathDate", "

    for(int i = 0; i<carriers.size();i++){
        Carrier carrier = carriers.get(i);
        carrier.setTopics(topics);
        String entity = carrier.getID();

        String query = "...";

        //Runs the query and adds the results to the Carrier.
        resultFormatter(query, carrier);


    }

    return carriers;

}
```

2. Initiate a list of the topics that is defined in the SPARQL query.
3. In a for loop iterating over all the carriers selected earlier (in the count function) add the SPARQL query. In the picture above in the spot where it says "…". The string entity should be added in to the query, often as the first argument in the RDF triplet.

4. Once the function is completed it needs to be called. In the Query class there is a list of if statements. Add an additional statement that controls the type and calls the function.



```
if (carrier.getID().equals("")){

} else if (carrier.getType().equals("dbo:Organisation")){
    add_info = o.additionalInfo(carrier);
} else if (carrier.getType().equals("dbo:Person")){
    add_info = p.additionalInfo(carrier);
} else if (carrier.getType().equals("dbo:Location")){
    add_info = l.additionalInfo(carrier);
} else if (carrier.getType().equals("dbo:Country")){
    add_info = l.additionalInfo(carrier);
}
```