# DBpedia Accessor to Check Query Type

Course: Introduction to Natural Language Processing (INLP)  
Authors: Andrei Mihai, Andrei Polzounov, Deividas Skiparis.  
Date: 2015/10/04

|  |  |  |
| --- | --- | --- |
| **Number of Queries:** | 473 |  |
| **Correct answers** | 449 | 94.93% |
| **Incorrect answers** | 24 | 5.07% |
| **Number of false positives:** | 8 | 1.78% |
| **Number of false negatives:** | 16 | 3.56% |

Table 1: Classification results

We addressed this problem by querying DBpedia database using a SPARQL endpoint. We chose this method, because SPARQL is specifically designed to work with RDF databases [[1]](#footnote-1) (which is DBpedia), so we expected the best results. It was observed that with several SPARQL queries we are able to determine quite accurately (95%, see Table 1: Classification results) whether a string could correspond to a location. To enable us to use this query language within the code, an add-on named SPARQLWrapper had to be downloaded. The Python code itself consists of 4 stages:

1. **Query Direct URI.**

In the first stage we check if a string has a dedicated DBpedia page. For example, “Berlin” has a page <http://dbpedia.org/resources/Berlin>. The code looks for exact string in ‘rdfs:label’ property throughout all DBpedia resources. If a search matches, a corresponding URI is returned, provided the resource is a page with information (not Category, Disambiguate, etc.).

1. **Check for Redirections**

Some pages in DBpedia are simply redirections to other pages. The redirection page itself does not hold any content, so it is important to find a parent page. The code tests if a string has a page in DBpedia and uses ‘dbo:wikiPageRedirects’ category to check if it redirects to a parent page and returns it´s URI, if it does. The advantages of this, is that redirection pages sometimes account for most common typing errors (e.g. Barcellona or Barcelone will refer to Barcelona page).

1. **Check if returned URI is a location**

If a URI to a resource has been returned from step 1 or 2, then we want to check the type of this resource. Type of a resource is stored under ‘rdf:type’ tag and the code examines if it has at least one of the tags, which describe location (e.g.: dbo:Place, dbo:Location). If it does, it means the string we are querying can be classed as [Location].

1. **Check for disambiguates**

If step 3 returns that a string is NOT a location, it is possible that we are dealing with a resource with disambiguates. For example <http://dbpedia.org/resource/Bluefield> opens a page with information, that “Bluefield” could be paraphrased as Bluefields, Bluefield,\_Virginia and Bluefield,\_West\_Virginia. So in order to check whether a string (e.g. “Bluefield”) is a location, the codes checks all disambiguates and determines if they are locations. If any of them matches a location, it is possible, that queried string (e.g. “Bluefield”) refers to a location.

This program is relatively reliable to determine if a keyword could refer to a location. However it has some limitations too:

1. This code is not tolerant to grammar errors (rare exceptions, as discussed in Step 2)
2. It will not work with strings with several locations. For example, searching “Barcelona” and “Spain” will return positive results. However “Barcelona, Spain” will return an error. Likewise, USA state abbreviations will not work either, unless written in full.
3. It will not return true results if pages on DBpedia have errors in them. For example, Glen\_Canyon is classed as an [Event]
4. If a queried string has disambiguates, it is not possible to be sure, if a user is searching for a location.

1. https://code.google.com/p/tdwg-rdf/wiki/Beginners6SPARQL [↑](#footnote-ref-1)