Introduction to Natural Language Processing: First Deliverable

Samuel Pedrajas and Xavier Serra

October 2015

1 Introduction

In this deliverable, we have developed a *DBpedia* interface in Python to allow the user to check if a string is a location, region, place, country, city, state or town.

2 Description of the problem faced

The problem faced in this deliverable was to be able to access the *DBpedia*. The *DBpedia* is an online ontology containing information about milions of words or concepts (such as "Los Angeles", "Spain"...). In order to be able to work more easily with it, we have had to develop an interface with which to do simple queries to the *DBpedia*, saving the final user all the trouble to establish a connection with it. Concretely, the interface had to be able to determine if a given *string* was considered a location, a region, a place, a country, a city, an state or a town (notice that more than one option can be simultaneously fulfilled).

3 Description of the approach

Due to the requirements of the deliverable, the implementation had to be done in **Python**. Fortunately, there is a library, *Sparqlwrapper*, which can easily communicate with *DBpedia*. Using it we were able to configure simple queries to *DBpedia* in order to check if a certain token belonged or not to a given category. With this implemented, we just developed a simple interface providing a method for each possible category. These methods receive a *string* and return a *boolean* determining whether the given *string* belongs or not to the corresponding category.

4 Files provided

Two files are provided:

- *DBPediaAccessor_classes.py:* This class provides all necessary methods to easily consult *DBpedia*. Its main methods, and the ones that should be used, are:

```
def is_place(self, entity_to_check):
    # Checks if entity_to_check is a place
    return self.ask_query(entity_to_check, "Place")

def is_location(self, entity_to_check):
    # Checks if entity_to_check is a location
    return self.ask_query(entity_to_check, "Location")
```

```
def is_country(self, entity_to_check):
    # Checks if entity_to_check is a country
    return self.ask_query(entity_to_check, "Country")

def is_city(self, entity_to_check):
    # Checks if entity_to_check is a city
    return self.ask_query(entity_to_check, "City")

def is_town(self, entity_to_check):
    # Checks if entity_to_check is a town
    return self.ask_query(entity_to_check, "Town")

def is_state(self, entity_to_check):
    # Checks if entity_to_check is a state
    return self.ask_query(entity_to_check, "State")

def is_region(self, entity_to_check):
    # Checks if entity_to_check is a region
    return self.ask_query(entity_to_check, "Region")
```

- *DBPediaAccessor.py:* This program provides an easy way to test the class *DBPediaAccessor_classes.py*. It receives a file name, and the desired category (state, city, ...) and it returns, for every line in the file, whether it belongs the the given category. Logically, this means that every line in the file must contain only one concept to search.

5 Evaluation of the results

In order to determine the goodness of the implementation obtained, some examples were provided by the teacher of the subject. These examples belonged to a certain, and known, category, so that we could compare the answer given by our program with the right solution.

This comparison has resulted in a 100% of accuracy. Therefore, we could conclude that we have been successful in our implementation