Practical Language Technology:

a Q/A System for DBpedia

with a Focus on The Olympic Games

Group 13: Xiaoying Chen(s2714140), Roald Baas(s1879642), Anne Wanningen(s2219832)

June 24, 2016

1 Introduction

Our task for the course was to build a Dutch question-answer system with the theme: Olympic Games. The Dutch DBpedia is used as a source of information in answering questions. DBpedia is a project which extracts machine readable information from Wikipedia, making it available for free as Linked Open Data. The information (which are objects, concepts, resources) on DBpedia are identified by a URI (which often follow the syntax of URL's), which can easily be accessed via an internet browser or using a query language such as SPARQL. It supports 111 different languages, of which the English version is the largest. In our Q/A system we limit ourselves to the Dutch version.

The system is evaluated using 50 test questions, which were unknown at the time of constructing the system. The questions were expected to be of the following types:

· 'Wie/wat is de/het X van Y?'-questions
· List-questions (Welke landen...)
· Count-questions (Hoeveel landen...)

The questions were also expected to have the following properties:

· Questions will be in Dutch
· Questions will not contain spelling errors
· Questions are about the Olympics
· No questions that require context
· No 'double' questions (Waar en in welk jaar...)
· No 'or' questions (Waren de Olympische Spelen in 2002 zomer of winter spelen?)

Ofcourse the questions may be phrased differently than in these types (for instance 'Door wie...' instead of 'Wie...'). This means that the system has to start by analyzing each question to extract the relevant information used to answer the question, which will be explained in more detail in the next section.

2 Architecture

The system was built in Python3 using the libraries lxml and SPARQLWrapper. We make use of the files *pairCounts* and *similarwords* provided by Spotlight (Daiber, Jakob, Hokamp, and Mendes, 2013) to determine dbpedia URI's and properties of questions. Alpino is used to parse questions, giving XML data as output. XPath is used to extract keywords from this XML data. Finally, after having analysed the question, SPARQL is used to query DBpedia, possibly returning an answer.

Since the system is limited by only answering questions about the Olympics in Dutch and due to the machine readable nature of DBpedia, the file pairCounts (the Dutch version, as other languages are available) can be used to determine URI's. The file similarwords gives us the ability to generate words similar to a given word (as the name suggests).

The input is a text file containing questions (including question number, separated by tabs). The system iterates over this file, analyzing and answering each question, giving the output in a text file using the same format: question number followed by the answer(s), separated by tabs.

The system uses two methods to try and find answers to questions: a fast method and a slower more thorough method.

2.1 Method 1

The first method starts by determining the question type ('wat', 'wie', 'waar', 'wanneer', 'hoeveel', 'welke', 'hoe'). There are slight individual differences between these questions. For example, if the question type 'hoe' also contains the word 'lang', we can add 'lengte' to the list of properties to check (meaning the system will find lengths of athletes).

The method then gets the property and the URI from which to find an answer. The subject of the question is used as a property, and the object is used to find the URI (using pairCounts). For example, in a 'Wie/wat is de/het X van Y?'-question, X is the subject, and Y the object. The property and URI are then used in a SPARQL query to find the information stored in the found property of the found URI.

Since this method is rather quick and dirty, it won't find many answers (but

those that it does, it finds quickly and accurately as we will see in the Results section). If this method fails to produce an answer, the system tries again using the second method.

2.2 Method 2

This method aims to match a semi-ordered list of keywords to a ordered list of URIs. As the method lacks an absolute measure to predict keyword or URI relevance in a systematic manner, the order in which keywords and URIs are processed is paramount to it's success.

Keyword Extraction

A list of keywords is composed of all nouns in the question and their respective lemma's. All nouns that were originally accompanied by adjectives are also added combined with those adjectives. Next all keywords found so far are matched with URIs as described in the next section, in order to find DB-pedia specific class names for existing concepts. Any class names found in this way are also added to the list. Finally some question specific keywords are added. Co-occurrence of the words *waar* and *geboren* for instance, result in the keywords *geboorteplaats* and *geboortestad* being added to the end of the keyword list, in order to find some peculiarly phrased properties.

URI Matching

There are four sources of information on which URI matching is performed; the object of the question, the subject, other noun phrases and the individual nouns. These sources are utilized one after the other, until an answer is produced. Possible URIs are found using the *paircounts* file. The list of matching URIs is then ordered by rating each URI using the following formula:

$$75 \parallel K_q \cap K_{uri} \parallel -10 \parallel K_{uri} \parallel - \parallel URL \parallel$$

In which K_q is the set of keywords extracted from the question, K_{uri} is the set of keywords as mentioned in *paircounts* for that URI, and URL representing the DB-pedia address for the URI. The parameters for this function were manually tuned for optimal matching on the available training data.

Finding the Answer

For every ordered list of URIs the top 50 results are queried for a list of all properties on that page. If a property label contains a word from the keyword list a query is fired to get its value. If this query returns an acceptable value, this is considered the answer.

3 Results

Of the 50 test questions, the system found possible answers for 41 questions. Only 15 of those were correct. M1(The fist method) had found 5 correct answers, and M2(The second method) 10. The table below describes the relevant statistics for both methods, as well as the complete system.

Overview	Precision	Recall	F-score
Method 1	0.83	0.1	0.18
Method 2	0.29	0.2	0.24
Combined	0.37	0.3	0.33

Interestingly, the combined system performs significantly better then both methods individually. Although M1 outperforms M2 in terms of precision, M2 performs better on recall. Both methods produce answers the other method is incapable of finding, and the precision of M1 is such that it is sufficiently reliable to execute the methods in a serial fashion and blindly accept the first answer. The apparent superiority of the combined system is further confirmed by it's F-score which is, for this data, substantially higher then both systems individually.

As mentioned before, the first method is faster. The mean time to answer a question in method 1 was 2.7(sd 1.2) seconds, whereas that of method 2 was 15.9(sd 20.7) seconds. This large increase in time is due to the system not being able to find an answer immediately, which is made apparent by the fact that if an answer was correctly answered, the mean time to answer is just 6 seconds (sd 3.4). In order to increase the speed of the system, we considered a timeout mechanism, but incidentally long processing times did produce correct answers. The longest processing time producing a correct answer in the test set was 14 seconds. Ultimately, using Binominal Logistic Regression on the test results, no significant covariance (p=0.051) between the processing time of M2 and whether the answer was correct was found.The questions answered correctly by the system were the following:

· [1] Wat is de geboorteplaats van Inge de Bruijn? (M2)
· [2] Wat is de geboortedatum van Leontien van Moorsel? (M1, M2)
· [3] Wat is de bijnaam van Leontien van Moorsel? (M1,M2)
· [7] Wat is de volledige naam van Mark Tuitert? (M2)
· [8] Wat is de specialisatie van Gerard van Velde? (M1, M2)
· [9] Op welke onderdelen kwam Nicolien Sauerbreij uit op de Olympische Spelen? (M2)
· [13] Door wie werden de Olympische Winterspelen 2014 geopend? (M2)
· [16] Wie opende de Paralympische Spelen 2012? (M2)
· [21] Wie is de voorzitter van het NOC*NSF? (M2)
· [22] Wat is de website van het NOC*NSF? (M2)
· [25] Wanneer beginnen de Olympische Zomerspelen van 2016? (M1)
· [29] Hoeveel landen deden mee aan de zomerspelen van 2000? (M2)

· [35] Wanneer werd Churandy Martina geboren? (M1)
· [45] Wat is het motto van de winterspelen in Sotsji? (M2)
· [48] Wie is de coach van het Nederlands hockeyteam? (M2)

Most of the questions for which our system found a correct answer were with the questions with the format: 'Wie/Wat is de/het X van Y' (questions 1, 2, 3, 7, 8, 21, 22, 45, 48). In order to find answers for this type basic questions, property and URI are only needed (as mentioned before). Finding the property (X) and URI (Y) in cases like this proved to be no problem, which resulted in a high performance for these questions.

Other notable answers we found were that of question 9 (which sports did someone participate in), questions concerning the opener of the Olympics (13 and 16), startdates (25), the number of participating countries (29) and birthdates (35). Question 45 deserves a special mention, since the system was able to determine the URI 'Olympische_Winterspelen_2014' based on the object 'winterspelen in Sotsji'.

3.1 Error analysis

The system failed to answer a lot of questions correctly. Some questions were asked in a different way which meant that the system could not answer them. Another question contained a spelling error, causing the system to fail to answer it correctly. We have identified the main points which could be improved in the system as follows:

1. One single type SPARQL-query for all questions
   There are questions that give a property and the value of that property, which ask for the URI. Take question 34 for example: 'Wiens bijnaam is Lightning Bolt?'. This is a question that our system could answer correctly, while it can actually be easy to answer. After Alpino has parsed the question and found the property and property value, we could search for the value 'Lightning Bolt' in the 'bijnaam' property in a list of Olympic athletes.

   In the first method, we only made a single SPARQL query to find an answer. This could have been made more robust by trying different queries instead of immediately switching to the second method. The first method could have had some expansions to more question types, and an increase in queries by adding a dictionary to try more than one property.

2. Difficulty on finding correct URI by method 1
   When the phrase 'Olympische Zomerspelen' or similar phrases appeared in a question, the system uses it as URI in the SPARQL-query.

This however is not always the correct URI for the query. For example, in question 10: 'Hoeveel Nederlandse sporters deden mee aan de Olympische Winterspelen van 2002?', in method 1, the system gets the URI 'Olympische_Winterspelen_2002', where it should be 'Nederland_op_de_Olympische_Winterspelen_2002'. When a country name is mentioned in the question, most of the time the correct URI should be in this format in order to find the correct answer. Our system did not perform this check. This automatically means questions of this type could not be answered correctly.

4 Division of labour

The division of work was equal. Every member of our group contributed to the construction and testing of the system, the presentation, and the report.

References

Dutch DBpedia. URL `http://nl.dbpedia.org`.

lxml. URL `http://lxml.de`.

Python3. URL `http://www.python.org`.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*, 2013.