Knowledge Graph from Swedish Wikipedia

Vanja Tufvesson
LTH, Lund University
Lund, Sweden
pi08vt1@student.lth.se

Thomas Hassan Resa
LTH, Lund University
Lund, Sweden
atf09tha@student.lu.se

Abstract

A question answering system was developed based on the answers from the game ”Kvitt eller Dubbelt”. The information was extracted, based on the answers, from Swedish Wikipeda and processed using powerful tools. The resulting system could take a query, search through a knowledge graph and present relevant results. The coverage of named entites from the question and the answer was not very high.

1 Introduction

Inspired by question answering systems like Google and IBM Watson, we wanted to create a system that could answer Swedish questions from the game ”Kvitt eller dubbelt”. Starting from the answer to a question, can we extract information from Wikipedia that would contain the question in ”Kvitt eller dubbelt”? In order to answer this question, we downloaded a dump of Swedish Wikipedia and used multiple freely available tools to extract the relevant content, tag, parse it and finally build a searchable knowledge graph.

2 Previous work

2.1 Watson

Watson (Ferrucci, 2012) is a question answering computer system developed by IBM. It was specifically developed to be able to answer questions on the quiz show Jeopardy. In 2011, the system beat the two highest ranked players in a Jeopardy match. (Alpman, 2014) The system relies on natural language processing, information retrieval, knowledge representation, automated reasoning and machine learning technologies.

IBM continues to invest in the super computer system, in order for it to delve into medical records and help doctors find the right diagnosis for their patients. The system is also being tested for customer service usage at companies.

2.2 Google Info Box

Google has started to structure some of the large amounts of data on the web. The search engine is now working as an informative, semantic search engine, providing an information box, instead of just links, as the result of a search query. The type of information varies but it usually contains some Wikipedia content. It can also contain a picture and/or a map. The box appears for querys like actors, artists, places and also for questions like ”what is the highest mountain in the world”. (Parfeni, 2012)

3 Pipeline

The project descirbed takes a wikipedia dump and produces a knowledge graph from specific named entities. The entire project is coded in Java 1.7 and uses several open-source applications to extract the contents from wikipedia and then present it in a web browser.

3.1 Index Creation

An index was built using a wikipedia dump and Apache Lucene.

3.1.1 Wikipedia content

The first step in order to create an index of the Swedish Wikipedia content was to download a text dump of the Swedish version of Wikipedia. The downloaded dump (Wikipedia dump, 2013) was all with all pages, current versions only from 2013-11-01. This dump was the latest found when the project was started. The number of articles on Swedish wikipedia has incresed with almost 67 % during 2013 and with a larger corpus would hopefully result in a larger knowledge graph. This dump contained XML-markup and this was re-

moved using Wikipedia Extractor (Wikipedia Extractor, 2014).

3.2 Lucene

An index was built using the open source library Apache Lucene (Apache Lucene, 4.6.0). Lucene is a text search engine library which uses ranked searching. Starting from the Lucene demos it was fairly straight forward to go through the contents of the Wikipedia dump and build the index.

3.3 Part of Speech Tagging

To reduce the size of the problem we only focused on cards with single word answers, which were named entities. To extract those named entities the first step was to tag the text content of the cards with their corresponding Part of Speech tag. We used Stagger - Stockholm Tagger (Stagger,) for the tagging since it was already trained for the Swedish language. Then we extracted all the named entities from the answers.

3.4 Index Search

In the next step we extracted relevant sentences for each named entity from the Lucene index. Just like the indexing, searching the Lucene index was also straight forward, starting from the demo code. The matching search results were tagged with their part of speech tags using Stagger and written to disk.

3.5 Triplets Creation

To create the subject-object-verb triplets, the next step was to create a dependency graph from the tagged results. This was done to extract knowledge from text to triplets.

3.5.1 Dependency Parsing

The tagged search results file was parsed using MaltParser (MaltParser, 1.7.2), a dependency parser developed at Vxj University and Uppsala University, Sweden. The parsing methodology is based on deterministic parsing for building labeled dependency graphs, history-based models for predicting the next parser action at nondeterministic choice points and discriminative learning to map histories to parser actions.

Maltparser came with a few different pretrained models. We used the Swedish model which was trained on the Swedish treebank Talbanken05.

3.5.2 Subject-Object-Verb triplets

From the dependency graph, we could excract triplets by looking through each sentence for different objects that were related to the same head. The subjects and objects and their realtionships were easy to identify by looking at the dependency graph created in the previous step.

3.6 Graph Storage

The triples extracted from the the corpus was stored with the help of OpenRDF Sesame (Sesame, 2.7.8). A main memory repository was created to store the triples. The reason why a main memory repository was used was because it was faster then having an online repository. Sesame's Java API was easy to use and uses the SPARQL language to query the graph.

4 Results

This section contains both the web interface which can query the knowledge graph and an evaluation of the coverage of the knowledge graph.

4.1 The webinterface

The web interface was constructed with Apache Tomcat (Tomcat, 7.0.42) and the Java Servlet API. It is a basic html page where the user can input a named entity and press search. This will display all the triples containing that named entity both as a subject and as an object. An example query of "Stockholm" is shown in figure 1.

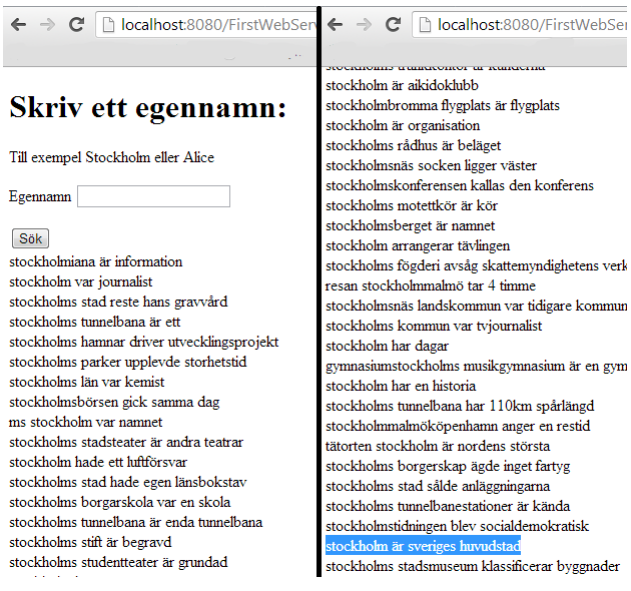

Figure 1: A screenshot of the web interface.

The highligthed text in figure 1 says: ”sverige är stockholms huvudstad”.

4.2 Evaluating the knowledge graph

In an attempt to evaluate how much knowledgde that was extracted from wikipedia, the questions and answers from ”Kvitt eller Dubbelt” were used. For every question and answer the named entities were extracted. Then if both the question and answer had named enitites (one or many) the knowledge graph was searched to see if these named entites had a link. If they had a link it counted as a hit and thereby it was possible to see how many questions that could be answered with the created knowledge graph.

The result is found in table 1:

Number Of Hits:	116.0
Total Number:	325.0
Procent coverage:	0.357

Table 1: The coverage of how many questions that could be answered by the knowledge graph.

This means that this project was able to extract the information needed to answer around 36 % of the questions were a named entity was the answer.

5 Discussion

There are several reasons why this knowledge graph obtained such a low procent coverage.

5.1 Creation of triples

When paragraphs were extracted from wikipedia regarding a certain named entity. All sentences were scanned according to the subject-verb-object and if these did not exist the triple were never created. So all of the facts found in a Wikipedia fact box are lost since those texts do not contain verbs.

Another problem is that in paragraphs we do not replace the pronouns with a previous named entity and thereby a lot of information is lost. Example sentence: ”Marko Lehtosalo fddes i Finland. Han gick i Myrsjskolan frn frsta till nionde klass.” It would have been smarter to replace the pronoun ”han” with the named entity ”Marko ”Lehtosalo”.

The previous example also takes up another problem were our search of paragraphs will not find the named entity ”Marko Lehtosalo” since we are searching for named entiteis found in answers of ”Kvitt eller Dubbelt” and named entity there is called ”Markoolio”.

5.2 Limitations of Swedish Wikipedia

Swedish Wikipedia is not as progressive as the Enlish Wikipedia. Swedish wikipedia has got 1 million articles and English has got 4.5 million. Also the information found on Swedish Wikipedia is not as elaborated as it is on the English counterpart. For example there is no wikipedia article on Jasmine (the disney princess) on Swedish Wikipedia (2014-01-13).

6 Possible improvements

This project was limited to only single worded named entites and it would yield a larger knowledge graph with named entites larger than one word.

To get better results with the present evaluation method the triples should have been created by looking at named entities in the sentence. These named entities should then be linked with an appropriate link for example the noun or verb. Example ”Stockholm är Sveriges huvudstad.” take the two named entites and link them with the noun.

References

Ferrucci, D.A., IBM Journal of Research and Development (Volume:56 , Issue: 3.4.), 2012.

NyTeknik, Marie Alpman
http://www.nyteknik.se/nyheter/it_telekom/datorer/article3796685.ece
10 januari 2014

Softpedia, Lucian Parfeni
http://news.softpedia.com/news/Google-Testing-Semantic-Search-Engines-Which-Provides-Answers-Not-Links-268645.shtml
10 Januari 2014

MaltParser 1.7.2
http://www.maltparser.org/
13 Januari 2014

Lucene 4.6.0
http://lucene.apache.org/
13 Januari 2014

Sesame 2.7.8
http://www.openrdf.org/
13 Januari 2014

Stagger
http://www.ling.su.se/english/nlp/tools/stagger/stagger-the-stockholm-tagger-1.98986
13 Januari 2014

Tomcat 7.0.42
http://tomcat.apache.org/
13 Januari 2014

Wikipedia dump
http://dumps.wikimedia.org/svwiki/20131101/svwiki-20131101-pages-meta-current.xml.bz2
13 Januari 2014

Wikipedia Extractor
http://medialab.di.unipi.it/wiki/Wikipedia_Extractor
13 Januari 2014