Towards a Swedish knowledge graph

Alfred Åkesson
ada10aak@student.lu.se

Jens Gustafsson
ada10jgu@student.lu.se

Victor Rugarn Strömberg
ada10vru@student.lu.se

Abstract

In this report, we present a way to extract data from the info-boxes in the Swedish Wikipedia. Our focus have been twofold. First of all, we have focused on the extraction of time-expressions, but we have built a system which can be extended for extraction of any property in the info-boxes. We have also started the work on an automatic ontology builder, since we found out that many properties in the info-boxes described the same thing.

1 Introduction

RDF is a standard model for structuring information on the web. It can be used to store triples of subject-predicate-object and link these between different sites, creating the semantic web.[1]

Wikipedia contains a huge amount of information and therefore several knowledge bases have been based on Wikipedia. Many of these knowledge bases are based on the English Wikipedia and very few are based on the Swedish Wikipedia. The only Swedish knowledge base we are aware of is the Knowledge Graph created by Google.[2]

Example of successful English knowledge bases, except from the knowledge graph created by Google, are DBpedia and YAGO2, which we will discuss more deeply in section 2.

The extraction framework used by DBpedia on the English Wikipedia has also been used on the Swedish Wikipedia. Sadly, the framework is not tailored for the Swedish Wikipedia, meaning that many properties, especially language- and region specific properties were incorrectly extracted. The reason to this is many. One of them being the rules for extraction witch are based on the English language and it is therefore unlikely to be able to extract data from the Swedish Wikipedia. Another reason is the representations. In Swedish these do not always match the representations in English. An example of this is the time representation. In Swedish the 24 hour time notation is used, while in the English language the AM-PM system is used. As seen, if the English time representation is used to parse a time in Swedish, it will most likely fail, the parser will at least not be able to parse any time expression which expresses the clock being past twelve (12 AM).

Wikipedia is open for anyone, meaning anybody can edit an existing article or create a new one. In Wikipedia, there exist no text-formatting rules, which means that the raw data of an article can vary, this forced us to create a system which can parse a wide range of different forms of formatted text. Nor are there any specific rules to follow when creating a property, this means that many different strings can be used to represent the same property in an info-box in Wikipedia. It was because of this, we found the need for an ontology for the Swedish Wikipedia, since it would result in an easier way of extracting the data.

A lot of work has been done in the creation of DBpedia's ontology, therefore our idea is to use their ontology in order to automatically create an ontology that can be used for the Swedish Wikipedia.

Our date extraction proved to be very accurate in our tests. The system measured an accuracy of 100%. We have not found any serious attempts to do the same thing for the Swedish Wikipedia to compare with. Our automatic ontology system got a recall of 80% and a precision of 93%. By comparing this to [6], who got a recall of f79% and precision of 87%, one can see that this is an improvement of prior results.

[1] http://www.w3.org/RDF/

[2] http://www.google.com/insidesearch/features/search/knowledge.html

2 Related work

2.1 DBpedia

DBpedia is a large-scale, multilingual knowledge base, which focuses on extracting structured information from Wikipedia in 111 different languages [4], and mapping these as RDF triples.

When extracting data, DBpedia uses two extraction methods: generic-based and mapping-based info-box extraction [1]. The generic-based extraction aims at achieving a wide coverage of properties, while the goal of the mapping-based extraction is to extract high quality data.

In the mapping-based extraction, strings in the info-boxes describing the same type of things are chunked together and considered to be the same property. This removes a lot of redundancy and makes the database easier to query. In order for DBpedia to do this, they manually mapped the 350 most common templates in Wikipedia to an ontology. This resulted in an ontology of classes, which contains 720 different properties [1].

2.2 YAGO2

YAGO2 is another example of a semantic knowledge base that is derived from Wikipedia. In addition to the RDF-triple, YAGO2 uses three more dimensions to describe the extracted data. Those dimensions are a geographical, a temporal and a keyword dimension [3]. The dimensions are represented by tags. This means that if a triple describes something which existed during a specific time period, the triple can be tagged with the tags *startsExistingOnDate* and *endsExistingAtDate*.

YAGO2 is based on its predecessor, YAGO and has been built with help of the taxonomy of WordNet, a lexical database, and by the use of about 100 manually defined relations. A lot of the extractions of the data from the info-boxes in Wikipedia are done with the help of regular expressions, i.e., if some part of the source text matches a specific pattern, a specific fact is extracted from the text. This is the same method we uses when we are extracting data from the info-boxes in our system.

2.3 Ontology matching

Exner and Nugues presented in 2012 a system which can be used for extraction of triples in the running text of Wikipedia articles [5].

They annotate each sentence in a Wikipedia article with predicate-argument-structures. From these structures, they generate all possible subject-predicate-object-triples. The subject and object in each triple is then matched against the existing triples in DBpedia, and if any predicate is found, they match the found predicate with the parsed one.

From here, they select the most matched predicate-pairs and consider them to be equal. This gives them the ability to then create new non-existing triples in DBpedia from the running text of the Wikipedia articles.

2.4 Automatic refining of the Wikipedia property-ontology

In a report written by Weld and Wu [6], a method to automatically refine the Wikipedia info-box ontology is described. In their system, they make use of the edit-history in the source code of a Wikipedia article.

Their idea (slightly simplified) is the following: If a string describing a specific property has been replaced with another string, and still has the same value, then it is likely that those two strings should be considered to describe the same thing. Therefore, they map all such strings to one single property in the Wikipedia ontology. Another refining method they use, is to look for attributes which are never used together in a template. If they never are used together, it is likely they describe the same property and are therefore considered to be equal. A third way they use to improve the ontology is to remove tailing numbers or other simple mutations of a string describing a property. In section 2, we will compare the results of our automatic property-ontology matcher with the results from this system.

2.5 Summary of related work

Our long term-goal is to create, or help others to create, a Swedish knowledge base similar to DBpedia and YAGO2. In order to achieve this, we must have an ontology that describes the relationships in the graphs. Therefore, we have used the two reports, described in section 2.3 and 2.4, to get inspiration for our work. We used the ideas about automatic ontology refining, described in 2.4 and used the idea about incorporate DBpedia in our ontology matching, described in 2.3.

3 Method

Before starting this section, we want to give you a short definition of four words, which will be used several times during the rest of this report. You can find the definition of each word in table 1.

Word	Meaning
Article	A Wikipedia article
Property	The name of a property in an info-box in an article
Value	The value of the property in an info-box
Context	All triples having the same type of article and type of value are considered to be in the same context

Table 1: Definition of common words in this article.

3.1 Applications used

We have used several different applications in the creation of this system. The following is a list of some of the applications/tools that we have used, which deserves a further explanation:

- Bliki Wikipedia Parsing Library
- Sweble
- OpenRDF - Sesame

The bliki engine is an XML-parser[3], which we are using in our system to parse XML-dumps of Wikipedia. Sweble is a mediawiki parser [2] which we have used in our system to parse and build an AST-tree of each Wikipedia article, such that we more easily can extract information from the info-boxes in Wikipedia. OpenRDF Sesame is a framework for processing RDF data[4], we are using Sesame in order to save our extracted triples and query them using SPARQL.

3.2 Proceedings

Our system can extract any type of property from an info-box. However, we have only implemented the extraction of properties describing when someone was born. In 3.2.2 we will give a description of how our date-extraction system works. In the section after that, section 3.2.3, we will give you a description of our ontology matcher works. But before we start those sections we will give you a general picture of how our system works, described in section 3.2.1.

[3]https://code.google.com/p/gwtwiki/

[4]www.openrdf.org

3.2.1 General Overview

First of all, we download a XML-dump of all the existing articles in the Swedish Wikipedia. We then use Bliki to extract each article as a text-string. We send this text-string in to Sweble and get an AST-tree[5] representation as output of the text-string. By traversing the tree, we can find the info-box, if there is any, from where we can extract the existing properties. To find a specific property and value in the tree, we are using regular expressions. Figure 1 describes the different steps in our system.

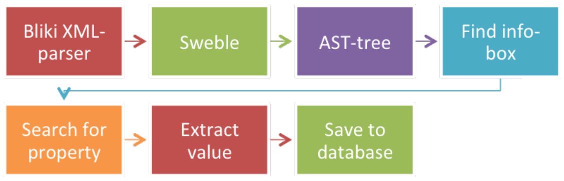

Figure 1: A picture describing how our system works

3.2.2 Date Matching

As mentioned in the introduction, a date can be represented in many different ways. Therefore, the first thing we did when we started our work on a date extraction was to decide how to represent the extracted dates. We choose to follow the XSD Date and Time Data Types, described by W3C[6]. Therefore, we save the extracted dates in the formats; Year-Month-Day (YYYY-MM-DD), Year-Month (YYYY-MM) and Year (YYYY), depending on how much information about the date we can extract.

As we just said, a date can be represented in many different ways, and therefore we had to create several regular expressions in order to be able to parse as many dates as possible. The most common date representations are given in table 2.

To find out what regular expressions to use, a lot of manually work had to be done. By looking in the source code of several different articles, we could find out what different ways date could

[5]Abstract-Syntax-Tree

[6]http://www.w3.org/TR/xmlschema11-2/

Date expressions
Year (YYYY)
Year-Month-Day (YYYY-MM-DD)
Day-Month-Year (DD-Month-YYYY)

Table 2: A description of the most common date-representations in the Swedish Wikipedia.

be represented in Wikipedia, and write regular expressions for these representations. Table 3 show the most common property used to describe when someone is born .

Common strings
Födelsedatum
Födelseår
Född
Född_datum
Född_år
Fdatum
föd

Table 3: A description of the most common strings used in the source-code of Swedish Wikipedia articles to describe when someone was born.

3.2.3 Ontology Matching

Figure 2 gives a general description of this part of the system. To create our ontology matcher, we used Sweble to extract all the properties in the info-boxes in Wikipedia which we temporarily saved in a SQLite database. After this, we started what we call the cleaning process. In this process, we removed all the triples which did not have a value linked to another article.

In Wikipedia articles often exist in many different languages and they are therefore linked together. DBpedia has extracted this interlinking data and that is something we took advantage of. We translated all the links in our database to the corresponding name in the English DBpedia.

The next step involves looking up the type of each article (subject) and value (object) in each triple and creating new triples which we saved to a file.

The new triples we created had the following shape: *(Type of Subject, Predicate(Property in Swedish), Type of Object)*. As described in the definition, we consider all the new triples having the same subject and predicate to be in the same context. Here is an example of one of the

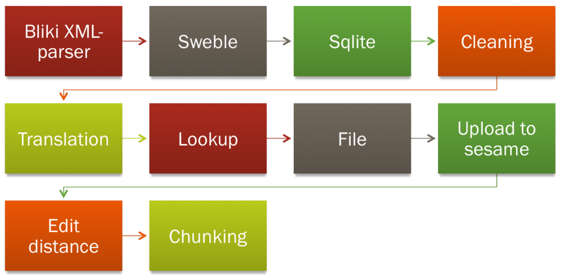

Figure 2: A figure describing our ontology matcher

triples we created: *(dbpedia.org/ontology/Place, kommun, dbpedia.org/ontology/PopulatedPlace)*

When this step was accomplished, we uploaded the file to Sesame. Finally, we calculated the edit distance between all properties in the same context and all properties which had an edit distance below a specific threshold where clustered together.

4 Results

4.1 Results: Date matching

To evaluate the date matching we selected 50 articles of persons at random that contained an info box containing a birth date. We then compared what we had in our system with the actual value of the birth date in the article. Of the 50 random articles, all dates were correctly extracted. If we look at the Swedish DBpedia we can see that date results are not comparable, for example:

```
<http://sv.dbpedia.org/resource/Astrid_Lindgren>
<http://sv.dbpedia.org/property/föddDatum>
<http://www.w3.org/2001/XMLSchema#integer>
```

4.2 Results: Automatic ontology

In order to evaluate the system, we have chosen three different contexts. The contexts we have chosen are the following:

- Place - Place
- Person - Time period
- Organization - Organization

In table 4 we show our selection of results we got for a system in different contexts. In the context Place-Place one can see that the two words stat, 'state' in English and stad, 'city' in english, have been chunked together, even though the meaning of the words are different. This is a problem with our system. Two properties that

Subject	Predicate	Object
dbpedia.org/ontology/Place	högsta_punkt (highest point) högstapunkt (highest point) hogstapunkt (highest point) — state state4 state5 state2 state3 state1 stat (state) stad (city) — Störstastad (largest city) största_stad (largest city) störstastad (largest city)	dbpedia.org/ontology/Place
dbpedia.org/ontology/Person	efterträdare (successor) efterträdare2 (successor) — maka (wife) make (husband) — grundat (based) grundad (based)	dbpedia.org/ontology/ TimePeriod
dbpedia.org/ontology/ Organisation	Gick_upp_i (merge) gick_upp_i (merge) — kammare2 (chamber) kammare1 (chamber) — internationell (international) internationellt (international)	dbpedia.org/ontology/ Organisation

Table 4: An selection of groupings when we used normal edit distance. Threshold 1.

have different meaning but are lexically very similar in the same context is not possible to differentiate from each other. However, if the contexts are different, it is possible to distinguish between the two words, this is due to the fact that we do not even try to match similar properties from different contexts together.

In the context of Person-TimePeriod one of the chunks extracted contained the two properties "make" and "maka", which in English translates to spouse(male and female). Depending on what you want to achieve with the reduction, this reduction can be seen as both successful and unsuccessful. Successful, because it was possible to map "make" and "maka" to mean the same thing, and unsuccessful if one do not want to consider those words to be the same. We consider those words to be the same and would in our ontology map these two strings with the property "gift med", married to, spouse.

Our system detects some inflections in the case of "grundat" and "grundad" and "internationell" and "internationellt". In table 5, the first row shows how large percentage of the predicates that our system reduced to another previous predicate. On the second row we show the same thing in actual numbers. The third row show how many of the chunks that contained false positives in relation to how many chunks there were containing more than one predicate.

In table 5 we see for the context Organization - Organization, we only reduced the context with 11 %. However, we obtained zero false positives. By manually going through the 119 properties in this context we concluded that a reason to why the reduction was smaller was due to the fact that the properties differed quite a lot from each other.

4.3 Evaluation

To get a picture of how accurate our automatic matching is, we have to compare it to a manual

	Place and Place	**Person and TimePeriod**	**Organisation and Organisation**
Reduced in percent	38 %	36 %	11 %
Reduced in number	118 of 310	36 of 101	13 of 119
Number of chunks larger than size one containing false positives	3 of 37	1 of 14	0 of 9

Table 5: Reductions and false positives

matching. Therefore, we chose to make a manual matching of one of the contexts. The context we chose was Person - Time period.

In this context, we managed to create 13 different chunks, while our automatic matcher created 14 chunks. Even though we did not extract the same number of chunks, the computer extracted the same chunks as we did. This means that there was only one chunk that the computer faulty created. The chunk the computer created which we did not was “Datum – fdatum”, which in english means “date - bdate”, where b means born.

9 of the chunks the computer created contained the exact same properties as our manually chunking did. This means that 9 out of 14 chunks the computer extracted were equal to the chunks we manually created. One of the chunks was 90 % equal to the chunk manually created, two of the chunks were 66 % equal and one 33 %.

Our manually reduction resulted in 36 reductions. Out of these, our system, made 29 correct reductions. This gave us a recall of 80 %, compared to [6] 79 %. If we sum the false positive in the reduction we get a precision of 93 %, compared to 87 % from [6].

5 Conclusion

The only Swedish knowledge base of today, which we are aware of and is accurate enough to actually be useful, is the *Knowledge Graph*, created by Google. However, this knowledge base is not public available and there exist no information about how it is created and what it actually contains. Therefore, we see a need for the development of a Swedish open knowledge base, and we like to believe that our work can be helpful in the creation of such a knowledge base.

We have created a date parser, which can parse when somebody is born with a very high accuracy. This parser can easily be extended to parse any kind of date property, for example when someone has died and when something has been created. Actually, our parser could be extended to parse other info-box properties as well, though we have not put any work into it.

Why we chose to focus on date extraction, is because dates are one of the most complex and regional dependent types of data you want to have in a knowledge base. Therefore, this work can be very useful in the creation of a Swedish knowledge base.

It is important to have a small ontology with not too many types of relations and where all the relations are unique. Otherwise it will be hard to use the knowledge base, since you will have to search for many types of relations every time you want to make a query. It is also very time-expensive to manually search for all the different strings that are used in Wikipedia to describe the same property, therefore, our ontology matching system is a tool we believe that will be very useful in future works on Swedish knowledge bases.

6 Future work

6.1 Date of born extraction

We found our date extractor to be very accurate. Future work will be to extend the system to parse any kind of dates in an info-box. To be able to parse dates describing time periods is also a subject for future work.

6.2 Automatic Ontology Matching

Future work will be to improve the automatic ontology algorithm. Some improvements can be to use a more sophisticated comparison algorithm instead of edit distances, e.g, using morphology and synonyms. The selection of a context can be automatic for a given predicate and can select a more unique context, in that way get a smaller selection of alternatives to chunk. Another thing that can be done is to try if the value is not a link to another Wikipedia article, then try parse it as a date, number, name, enumeration or string to get the type of more properties. For example *population* that is not a property that can be chunked in our system because the value is a number.

References

[1] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.

[2] Hannes Dohrn and Dirk Riehle. Design and implementation of the sweble wikitext parser: Unlocking the structure within wikipedia. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, 2011.

[3] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web*, pages 229–232. ACM, 2011.

[4] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 2013.

[5] Nugues Pierre. Ontology matching: from propbank to dbpedia. 2012.

[6] Fei Wu and Daniel S Weld. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th international conference on World Wide Web*, pages 635–644. ACM, 2008.