Collecting Data about People from Wikipedia

Rasmus Ansin, Guisong Fu, Didrik Lundberg,
Christoffer Janson, Andreas Serra, Daniel Swensson

Royal Institute of Technology
ransin@kth.se guisong@kth.se didrikl@kth.se
cjanson@kth.se aserra@kth.se dsw@kth.se

Abstract

Creating and maintaining databases with structured information is a subject which has recently has gained increasing attention from both commercial and scientific points of view. We can access an increasing amount of information freely online, which is why parsing and structuring this information is such a fast-growing field. Here, we will discuss the possibilities of creating a parser for biographical Wikipedia infoboxes, our implementation, and our experimental results (a structured database with information 290549 *persons).*

I. Introduction

The objective of our research has been to retrieve as much information as possible regarding as many persons as possible from Wikipedia and produce a database (in our case, a JSON file) containing all the information in a standardised form. This includes finding broken/misspelled code on Wikipedia and correct them for the JSON file as well as parsing and removing Wiki markup to make the information presentable.

The technique of turning unstructured information found on the Internet to structured information in a database is known as "web scraping". The input is usually in the form of HTML code, although in our case we will access the Wiki markup code of Wikipedia articles directly. For simplicity, we will focus on biographical information only. On the English Wikipedia, biographical information is stored in a semi-structured format in *infoboxes* (often appearing to a user viewing Wikipedia in a web browser at the top right of an article). An infobox holds information about several *attributes*, which are separated by a '=" sign into an *attribute key* and an *attribute value*. For example, one row in the infoboxes might be

```
|birth_date={{bda|1971|6|28}}
```

where "birth_date" is the attribute key, and "{{bda|1971|6|28}}" is the attribute value. Running this through our value parser, we will get the string "28 June 1971" to store in our database as the birth date of the person in the current article. In Section III, we will describe each step in detail.

There are no hard barriers to making databases about other things such as organizations or animals, or even to make a combined database. A knowledge base such as ours but also containing extensive data about various entities and concepts like companies, geographical locations and products can be used effectively when analysing on-line news articles. For instance, a knowledge base can help you identify concepts an article talks about, and identify trends and connections among these (as done by Meltwater) or for knowledge-based reasoning (as done by Wolfram Alpha).

We have programmed simple search application for our database in the form of a web server as a proof of concept of the validity of our methods and as a stepping stone to future applications. The database is pre-generated in Python and Java programs. The entire project, and more in-depth technical descriptions, can

be found at the `git` repository ran4argus.

II. Related work

There are a number of projects which have aims similar to ours.

I. Wikidata

The Wikimedia sister project Wikidata [4] has the purpose of gathering as much information as possible about Wikipedia entities and store it in a structured and organized database. The information is stored as statements, with each statement containing a property with associated data values.

II. DBpedia

DBpedia is a crowd-sourced project which aims to extract structured information from Wikipedia and make it available on the web [1]. The DBpedia knowledge base can be used to construct complex queries combining multiple facts. However, the quality of DBpedia dumps is often accused of not holding a very high quality.

III. Wiki Markup Parsers

There are in fact a large amount of projects of varying quality which aim to parse different forms of Wikipedia information - either from HTML or from an XML dump. Many of them also output the result into HTML as opposed to the AST or json formats - their purpose is not to create a database but to display Web pages created from the Wiki markup.

Sweble Wikitext Parser is one of the most well known, an open source parser designed to parse the original MediaWiki Wikitext grammar to generate an abstract syntax tree (AST) [2]. The purpose of the AST format is to enable easy machine processing of the information.

III. Method

I. Identification of infoboxes

Starting from a Wikipedia XML dump, our first step was to identify biographical articles. On English Wikipedia, this is surprisingly easy because of the (soon to be deprecated) Persondata info, a section not featured in the HTML code which appears at the bottom of the Wiki markup code in all biographical articles (at least we have found no English biographical articles that lack it).

Our method simply grabs the first infobox (if any) in articles containing Persondata sections. We have not encountered any "false positive" infoboxes belonging to non-biographical articles. Our method encounters different challenges in different languages: German Wikipedia has the Persondata sections, but no biographical infoboxes. Russian and Swedish Wikipedia have no Persondata sections, and Wiki markup language in their respective languages (with the Russian one in Cyrillic script). In the absence of Persondata boxes you have to look at infobox types, attributes, categories or the text body to decide whether the infobox is biographical or not. Identification of infoboxes is thus the only part of the program that potentially depends on knowledge of languages (parsing of attribute values might be different if special environments are defined in non-English languages, but you do not require knowledge about what the words mean to parse them).

After biographical infoboxes have been extracted from the Wikipedia dump, they are fed to both the attribute key parser and the attribute value parser. In the case of Wiki markup list environments (which frequently occur on several rows), all the list rows are concatenated to a single string before they are sent to the value parser (for recognized list formats).

II. Attribute key standardisation

Many attribute keys have several slightly different key names, for example "birthdate" and "birth_date". This makes the data harder to

interpret, and to search. To solve this and standardise the keys in the JSON file we wrote the program (`java_key_cleaner`). This program takes a text file containing a list of all keys sorted according to the frequency of occurrence in the JSON file. Here, we make the assumption that most people who edit Wikipedia uses an established standard and know how to spell the words correctly - the "right standard" is simply defined as the most common version. The keys are then stripped of special tokens such as "|" or "-" and whitespaces. Then, starting from the most common key, the Levenshtein edit distance [3] from that key to each key less common than key is calculated. If the edit distance is less than two it is assumed that the less common key is misspelled, and it is consequently replaced by the more common variant of that key. This method makes the linguistically interesting assumptions that spelling mistakes are typically small in edit distance, and that attribute keys are not typically close to each other (one could imagine a strange language where the three most common attribute keys are "aba", "apa" and "ada"). Typically words with different meaning are rather evenly spread out in the Levenshtein space, and these assumptions hold impressively well.

birth_date
birth_dat
birthdate
| birth_date
"'birth_date"'
birth date
birth_dates

Table 1: *Examples of attribute keys that were merged into "birthdate" by our method.*

A threshold edit distance for merging attribute keys is required to prevent false positives in the edit distance comparison (for example "birth_palce" $\rightarrow$ "birth_name"). Keys ending with a number are also excluded from being merged, since Wikipedia syntax states that keys which ends with a number are reserved for separate instances of the same key (for example "successor1", "successor2", when these refer to successors to different offices).

III. Parsing of attribute values

The parsing of attribute values takes place in a very sequential fashion, where different objects and environments in Wiki markup language are removed in an order of how much internal data they hold, and - in the case of environments - the probability to have more internal nested environments. Regex matching is employed to identify the occurrence of patterns. First, you remove Wiki markup fragments which contain no useful information, or replace those can be replaced easily. Examples are "{{mdash}}" (replaced with a dash) and the {{small|example}} environment (removed, but the text "example" kept). No problems with nested or multiple statements exist at this level.

Then, you handle the slightly more complicated environments. One example is the Wiki markup environment for marriages. It contains one name (the person to whom the article subject was married), one date (the date of marriage) and optionally another date (the date of death or divorce), potentially accompanied by the cause of the end of the marriage. This is in addition to optional style format parameters. The challenge when parsing lies in extracting all the useful information (and discarding the rest), keeping track of what is what, and writing it to the database in a legible way. On this level nested or multiple statements can cause problems if you are not careful - doing away with the "simple" Wiki markup code helps with this. To give a specific example, "{{Marriage|Justine Musk|2000|2008|}}" would become "Justine Musk (m. 2000-2008)".

Lastly, you do the Wiki markup environments which are commonly found at the top level. These are typically Wiki markup lists. These contain an arbitrary number of elements, some style parameters, and are sometimes found multiple times in one entry, and rarely nested in one another. Our parser has support

for multiple list environments in one value and support for the most common combinations of nested lists. These (and some certain common combinations of lists separated "manually" by line breaks or middle bullets) will be returned as lists to the JSON database. For example, "{{Plainlist|* ETH Zurich* University of Zurich}}" will become the list "ETH Zurich", "University of Zurich" to be stored in the JSON database.

We have also made experimental efforts with getting structured information from categories. We are only able to use categories of special forms, though. For example, "American people of Irish descent" would add "American" to the "nationality" attribute key (if it is not already there), and "Irish" to "ethnicity" (again, if there is not already an identical entry).

In addition to this, there are numerous heuristical rules to account for different de facto standards in Wikipedia formatting. For example, list entries entirely enclosed in parantheses will be appended to the previous list entries, since these are in all cases we have seen comments on the former.

IV. Results

I. Attribute key standardisation

To test `java_key_cleaner`, some tests were run on a subset of the 66602 persons found in the file with raw data. This subset contains 6191 unique attribute keys and a total of 1476068 keys, making it an average of 22.16 keys per person. We recall our assumption (or in Wittensteiń's picture theory of language, indeed a definition) that the most common version of a key is the one that is correctly formatted and correctly spelled. When we restricted the program to comparing edit distances of keys with a maximum of 9 characters (for performance) 9.59% of the unique keys were merged into another, and so 2.67% of all keys in the JSON file were corrected. Examples of the changes can be seen in Table 1. If the restriction on the number of characters in a key (the *cut-off word length*) was increased to 12, a total of 10.64% were identified as misspelled or wrongly formatted. 3.22% of all keys in the JSON file were corrected to their correct forms. The increase in erroneous keys detected however came at the expense of time, with a running time increase due to the additional calculations of edit distances between the larger keys resulting in a significant increase in execution time (see Figure 2). The number of erroneous keys found with different restrictions to character length can be seen in Figure 1.

When the key cleaner was run on all persons in the JSON file (290549 persons) a total of 12365 unique keys were checked. Of these keys, 1834 occurred more than 100 times in Wikipedia and were used to correct keys which were misspelled or in an erroneous format. With the cut-off key length of 12 characters, 1608 keys were changed which makes approximately 13% of all unique keys on Wikipedia misspelled or written in a non-standard format. 195129 keys were changed in the JSON file which is 2.98% of all Wikipedia keys.

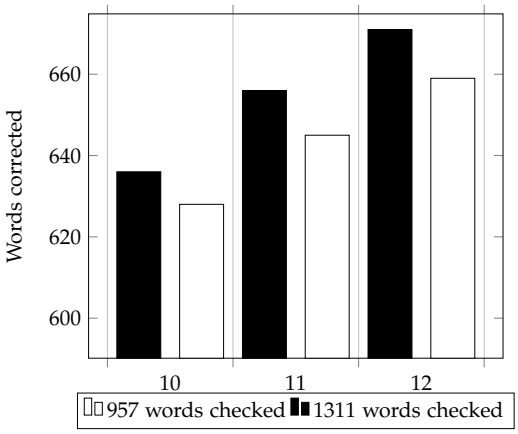

Figure 1: *Number of words corrected as a function of key length threshold.*

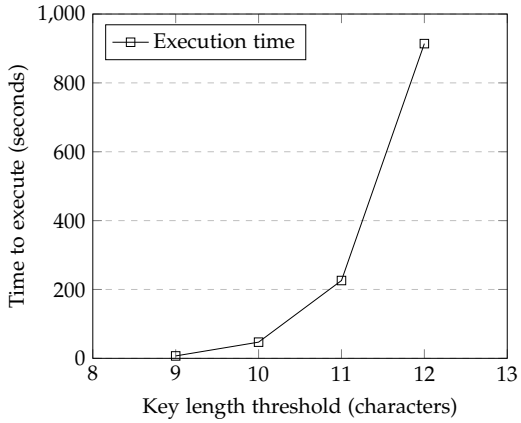

Figure 2: *Execution time of* `java_key_cleaner` *as a function of key length threshold.*

II. Parsing of attribute values

The parsing of attribute values is laborious to benchmark, since it requires a human to verify each attribute value. A study of 10 randomly chosen articles gives a precision - or positive predictive value - of 99.92% (where we compare our values with the Wikipedia values). This is a very high value, but we also typically throw away everything which fails to parse at several points (this was at first more of a consequence of how we happened to create our program rather than a deliberate strategy). Similarly, 5 randomly chosen Wikipedia articles give a very high recall (when we compare the infobox entries with our database values) of around 97%. We lack exactly similar projects to compare to, but our programs looks very good considering that there are still more Wiki markup templates we do not handle yet.

As a side note, special types of persons such as athletes can have massive amounts of infobox attributes, which are typically dependent on which sports they are active in. The database entries of these people can often appear very confusing. It is the strong suspicion of the authors that most of the erroneously parsed values can be found concentrated in a few articles.

V. Conclusions

A majority of the attribute keys on Wikipedia are correctly formatted. We were only able to confirm 2.98% of all keys as misspelled or of a non-standard format, which makes an average of 0.7 erroneous keys per biographical Wikipedia infobox. About 13% of all unique Wikipedia keys are misspelled or do not follow the standard format, which as previously stated is defined by deviation from the most common key format. 13% might sound like a significant part, but for most of these keys, the occurrence frequency over all our data are in the single or double digits, which still leaves 97% of the keys correct.

Unwittingly, writing a more strict (or more unstable, depending on how you look at it) parser of Wiki markup language than the one Wikipedia uses created a formidable tool for correcting formatting errors in infoboxes. Our spelling correction for attribute keys (and simply checking for rare occurrences) allowed us to find formatting errors and correct them. Similarly, the attribute value parser would output values containing Wiki markup language it could not parse and then output these, which allowed us both to see which environments we should next add parser support for and to identify infoboxes with bad formatting. Our program can thus also be used as a tool for improving Wikipedia.

Our method of parsing attribute values was surprisingly fast, and could handle a very large number of values correctly. Due to most articles being small and easy to parse, we also have a large number of correct complete database entries. It is often the people with the longer, more complicated infoboxes that are hard to parse though. So you have to take that into consideration as well.

We were able to extract 290549 persons from the Wikipedia dump. It is a difficult task to determine if these are all the persons that have a page on (the English version of) Wikipedia, but as far as we know, they are. The biographical information extracted by our program contains 6540761 lines of data divided over 10757

unique categories. 1482 of the unique categories have more than 100 entries meaning that 9275 categories occur in less than 100 Wikipedia pages. Some of these are probably misspelled, but unchanged by our program due to the fact that the edit difference between them and the correct attribute keys were too large. You can probably opt for hard-coding a merge between these as desired.

References

[1] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[2] Dohrn, H., and Riehle, D. Design and implementation of the sweble wikitext parser: unlocking the structured data of wikipedia. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration* (2011), ACM, pp. 72–81.

[3] Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (1966), vol. 10, pp. 707–710.

[4] Vrandečić, D., and Krötzsch, M. Wikidata: a free collaborative knowledgebase. *Communications of the ACM 57*, 10 (2014), 78–85.