Processing and linking address data

Summary of hands-on topic in Enterprise software seminar

Madis Nõmme, Karl Kilgi

Tartu
2013

Overview

In this task we built an address matching engine. The solution enables querying and displaying results of data which originating from a proprietary database, re-structures.

Source data

The source data originates from Estonian Road Administration. The source data was in CSV format with 38 columns. Data was denormalized and contained redundancies. To help with querying, we imported the data into relational database. This would provide useful for normalizing data. Considering the time constraints we only partially normalized the data. The import script is provided with the source code[1] of this project.

At the time of writing this summary, direct links to the source data have been removed but the data for Tallinn are available from the github repository given in references.

Solution

As a solution we developed a web service, supported by Ruby on Rails backend. The final implementation contains two implementations of the search.

Solution 1: Simple SQL search

Our first attempt to solve the problem was to use SQL querying capabilities provided by PostgreSQL database. We used the combination column TAISAADRESS as a column to make case-insensitive (ILIKE) WHERE conditions. If the query contained multiple words, we splitted it into keywords and AND'ed them together. This was done avoid omitting results because of user searching for example for 'Raekoja 33' but the searched column containing 'Raekoja plats 33'.

Solution 2: Full text search using Elasticsearch[3]

Elasticsearch is service that provides RESTful API to enable full text search[4]. Elasticsearch internally uses Apache Lucene[5]. Both of them are written in Java. To interface with Elasticsearch we used Tire[6], which provides Ruby interface for communicating with Elasticsearch API.

The way Lucene works on the principle of splitting all documents into words and building index for every word. Using index, which is exact string-match, makes lookup very fast. In our case the

TAISAADRESS column served as the document.

Using Elasticsearch (and thus Lucene) allowed us to make the search very fast while still maintaining accurate results.

Results

With Simple SQL solution the number of possible matches was sometimes very high. The number increased when making the query word more general. With each more general administrative unit Implementing a filter would have required parsing and analysing the query string to figure out (possibly based on some heuristic) to which level each search word belonged. To make the system respond in reasonable time with more general search terms, we limited the search results to 100 for the simple SQL solution

With full text search, searches were in very fast (~100 ms). The time did not increase with more general keywords. This was because of Lucene's indexing capabilities and using them when searching. Using Elasticsearch also allows assigning weights to certain keywords to make results more precise (e.g. reducing weight of tn (street) or mnt (highway) to make them less important in search).

Conclusion

We were able to replicate similar results to the ones provided by Maanteeamet's search[7]. The solution using Elasticsearch was much faster and also allowed easy adjustment of weights on certain query terms to make results more accurate. Both solutions could be made more precise and closer to the result user was actually searching for if some semantic analysis would be done to the query string (e.g. guessing whether a word was a street name and decreasing the importance of the results that do not have this word as a street name).
The authors found it surprising that full text search which is mainly meant for searching text documents, could be used for searching addresses (structured data) with relatively good results.

References

1. Source code of the solution	https://github.com/madis/ses-2013
2. Estonian Road Administration (Maanteeamet)	http://www.mnt.ee/?lang=en
3. Elasticsearch	http://www.elasticsearch.org/
4. Full text search	http://en.wikipedia.org/wiki/Full_text_search
5. Apache Lucene	http://lucene.apache.org/
6. Tire	https://github.com/karmi/tire
7. Maanteeamet's address search	http://xgis.maaamet.ee/adsavalik/ads