Delft University of Technology

Bachelor Graduation Project

Initial Research Report

UrbanSearch

Authors:
Tom Brunnik
Marko Malis
Gijs Reichert
Piet van Agtmaal

Supervisor:
Claudia Hauff
Clients:
Evert Meijers
Antoine Peris

April 26, 2017

TU Delft

Abstract

It is yet to be discovered how the importance of cities in the global network can be elucidated. In this document, the requirements for a solution to this problem are identified, as well as possible issues that might arise. We explore various methods and select the most appropriate ones that, if combined, satisfy these requirements and tackle the identified issues, in preparation for an attempt at solving the problem.

Keywords: urban, network, data mining, document analysis, filtering

Contents

1 Introduction

Common belief is that agglomeration benefits are key to economic growth [13]. However it may be that this economic growth's primary cause is the increase in (inter)national network embeddedness.

The huge amount of textual data generated online or the numeric historic archives are great sources of information on social and economic behaviours and constitute the bulk of information flowing among each other. Advanced text mining on newspapers and websites containing city names would allow to better understand the role of information in shaping urban systems.
Similar to research efforts in other domains such as financial trade [14] and sales forecasting [17], the idea is to develop search queries that capture urban-urban interactions as they can be found on the web through the co-occurrence of geographical names on websites e.g. "Zeeuws-Vlaanderen + Amsterdam " OR "Amsterdam + Zeeuws-Vlaanderen".

We will start by looking at related work that has been done on data from search queries, discussing the similarities to our research. Next we will further analyse the problem in its four main subsections: the extraction of data from the internet; the filtering and categorising of this data; the development of searchable queries and last the visualisation of the found data. From this problem analysis we will draw the requirement analysis. The requirement analysis is then used to decide upon the framework and tools we are going to use. And last we will give a short conclusion.

2 Related Work

3 Requirement Analysis

3.1 Must haves

1. General
   (a) Adding city names
   (b) Grouping relations and "zooming" on these relations
2. Search Engine
   (a) Filter results
   (b) Data mining
3. Filtering
   (a) Logic Filters
   (b) Relations Filters
4. Machine Learning
   (a) Types of relations
5. Visualization
   (a) Statistics of relations? Query relations
   (b) Strength of relations
   (c) Types: ML CBS defined

3.2 Should haves

1. General
   (a) Pluggable datasets
2. Machine Learning
   (a) Generalising relations, grouping relations

3.3 Could haves

1. General
   (a) International city names
2. Visualization
   (a) Front end for the app

3.4 Would haves

4 Framework and Tools

In this section, we evaluate what data sources are useful. Additionally, we discuss several methods and tools that can be helpful in storing and ingesting data. Furthermore, we describe numerous methods to filter and classify textual data. Then, we elaborate on different methods to perform queries with. We conclude with an overview of the available visualisation tools that can be used for displaying the results of the analysis.

4.1 Gathering the Data

4.1.1 Common Crawl

Common Crawl [2] is a freely accessible corpus of the pages across the web. Their data are updated and released on a monthly basis. Many researchers have used the data for varying purposes [16] [9] [15]. Since the project requires us to crawl the web (see section FIXME), the corpus is a very suitable candidate for us to work with.

The data of Common Crawl come in three formats[1]:

WARC This is the default and most verbose format. It stores the HTTP-response, information about the request and meta-data on the crawl process itself. The content is stored as HTML-content.

WAT Files of this type contain important meta-data, such as link addresses, about the WARC-records. This meta-data is computed for each of the three types of records (meta-data, request, and response). The textual content of the page is not present in this format.

WET This format only contains extracted plain text. No HTML tags are present in this text. For our purposes, this is the most useful format.

For extracting data from Common Crawl, many open-source libraries are available. Common Crawl's official website refers to `cdx-index-client`[2] as a command line interface to their data. It allows for, among others, specifying which dataset to use, supports multiple output formats (plain text, gzip or JSON) and can run in parallel.

[1] `https://gist.github.com/Smerity/e750f0ef0ab9aa366558`
[2] `https://github.com/ikreymer/cdx-index-client`

A simple query on the latest index using the online interface[3] yields 1676 pages of 15000 entries each, which are roughly 25 million entries in total. However, there are over 5.5 million registered domain names with top level domain `.nl`[4]. One would expect many more pages to exist with that number of domains. There are several explanations for this, including:

- Common large websites, such as `www.google.nl` and `www.wikipedia.nl` have not been fully indexed by Common Crawl, because their "parents", `www.google.com` and `www.wikipedia.org` have already been indexed almost entirely.
- Not every website allows their pages to be crawled. According to Common Crawl's official website, their bots can be blocked via the common `robots.txt`. Additionally, they honour so-called `no-follow` attributes that prevents the crawler to follow embedded links. Sites that use these features are therefore partially or not at all included in the indices of Common Crawl.

4.1.2 Delpher

Delpher [1] is a library consisting of millions of Dutch digitalised newspapers, books and magazines of ages differing from the fifteenth century up until now. Their data are free to use for private purposes. All data can be accessed however, if the organisation doing research explicitly asks for permission to use the data and agrees to Delpher's terms. Delpher could be useful to perform historical analysis and see trends over longer periods of time. This cannot be done using web pages as source, as the Internet is still very young.

Should historical analysis be desired, Delpher is the data source to use. Since their data are texts from reliable sources, there is no noise to be filtered out. Additionally, some valuable information can be extracted, such as area of distribution, cross-location referencing and target audience. These aid in the process of measuring the strengths of relationships.

4.1.3 Eurostat

Eurostat is the statistical office of the European Union situated in Luxembourg. Its mission is to provide high quality statistics for Europe [5]. We identified Eurostat as a source that is not useful for this particular problem. Although Eurostat contains a lot of statistics on European cities, there is not enough useful information which contributes to giving more insight into the network connectivity of cities. Therefore, we did not include Eurostat as an information source.

4.2 Data Storage and Ingestion

4.2.1 Elasticsearch

Elasticsearch is an open-source search engine which centrally stores your data [3]. It is a fast and scalable solution that was designed with big data search in mind. According to Kononenko et al. [7] Elasticsearch has some significant advantages in comparison with traditional relational databases. Two of these advantages are scalability and performance.

Scalability According to Elasticsearch [3] their product has no problem with scaling horizontally. It automatically manages indices and queries distributed across a cluster. This is an important feature as it is likely that the amount of data that our solution will use and process will increase and we do not want to keep upgrading the server that contains database, which would be vertical scalability. FIXME: Referentie naar dat we alleen NL doen nu? En daarom scalability nogal belangrijk is

[3] `http://index.commoncrawl.org/CC-MAIN-2017-13-index?url=*.nl&output=json&showNumPages=true`
[4] `https://www.sidn.nl/a/knowledge-and-development/statistics?language_id=2`

Performance Because Elasticsearch was designed to handle documents and perform full-text search it not surprising it performs well doing this. As we're going to be using the same kind of input data we expect Elasticsearch as the most choice. Kononenko et al. found that while scalability and schema-free documents are common for NoSQL systems, the combination of all three (scalability, agility, and performance) in one system is what makes Elasticsearch stand out from other systems. Following this, we conclude Elasticsearch would be a good choice as a data storage and search platform for our FIXME: solution—product—project.. (We moeten even kiezen welke we aanhouden zodat we dat overal hetzelfde doen.

A downside of Elasticsearch is that it does not have any form of security out of the box. This means that everyone with the server address could access the data. This is not a problem for CommonCrawl data, as this was already availbale online anyway. However, when using for example Delpher or other sources you need a license this becomes a problem. Next to that, it would also be possible for anyone to meddle with the data in Elasticsearch making the data unreliable. Elasticsearch provides a Security package for which you unfortunately need a paid license. However, to secure Elasticsearch while making it available for users we could use a plugin such as Search Guard [5] or use a special proxy as proposed by Kononenko et al. [7].

4.2.2 Hadoop

4.2.3 MrJob

4.2.4 Neo4j

Neo4j is a highly scalable native graph database that leverages data relationships as first-class entities [10], enabling enterprises of any size to connect their data and use the relationships to improve their businesses. Holzschuher and Peinl compared the performance of Neo4j to the more classic and commonly used NoSQL databases and found out that the more natural representation of relationships resulted in significant performance increase gains [6]. It is used by, among others, ebay, Cisco, Walmart, HP and LinkedIn[6]. TODO: expand

4.3 Filtering and Classifying

4.3.1 Clustering

4.3.2 Filtering

4.3.3 Machine Learning for classification

Machine learning for website classification works by using the raw text from a website. Because the text from websites is often unstructured the 'bag of words' model is used. This model counts how often each word is used. There are 2 libraries available which contain most steps needed. There is scikit-learn [8] for Python and Weka [11] for Java. The machine learning works in 4 steps:

1. **Creating a feature extractor**
   Given text from a website, returns the "features" from this text. Features are the words that occur in the text and the number of occurrences. Before this data is extracted stop words (the, is, at etc) are removed and the rest of the words are stemmed meaning all words will be changed to their root-forms (features - feature, controlled - control). The program Snowball [12] and NLKT [4] (which uses the snowball version) have a dutch implementation for this, although it might need to be improved a bit.

   Afterwards, to prepare the features for the machine learning algorithm. We need to give each feature a numeric id. Count each of these tokens. And we need to normalise the tokens. For this scikit-learn provides algorithms.

[5] `http://floragunn.com/searchguard/`
[6] `https://neo4j.com/customers/`

2. **Manually labelling**
   The first set of examples (which for each class can be just a few of websites), will be manually labelled.

3. **Generating a classifier**
   The labelled examples are fed to a learning algorithms. This will generate a classifier. Several algorithms (provided by scikit-learn) are:

   (a) SVM - LibSVM ($O(n^3)$?)/*LibLinear*

   (b) Naive Bayes

   (c) Neural Networks

   (d) Descision trees (usually C4.5)

4. **Entering new examples**
   When a new (unlabelled) example (website) comes - extract the features and feed it to your classifier - it will tell you what it thinks it is (and usually - what is the probability the classifier is correct). Afterwards the classifier is updated to include new features extracted from the example.

todos:
n-grams / shingling

4.3.4 TF-IDF

basic idea: 1. using training data to assign values on words - filter meaningless words - assign words with highest value as categories? 2. Do the same on training data for each category (choose a few documents manually per category) and then check for websites for which categories has the highest value.

4.4 Search Queries

4.4.1 Enter Queries

4.4.2 Get Results

4.4.3 Specifications

4.5 Visualisation

4.5.1 neo4j?

4.5.2 Connection between cities

4.5.3 The Strength of these connections

5 Conclusion

References

[1] Koninklijke Bibliotheek. Delpher - boeken kranten tijdschriften. `http://delpher.nl/`. Accessed: 2017-04-26.

[2] Common Crawl. Common crawl. `https://commoncrawl.org/`, 2017. Accessed: 2017-04-25.

[3] Elasticsearch. Elasticsearch. `https://www.elastic.co/products/elasticsearch`. Accessed: 2017-04-26.

[4] Alex Rudnick et al. Nlkt. http://www.nltk.org/api/nltk.stem.html, 2017.

[5] Eurostat: About. http://ec.europa.eu/eurostat/about/overview. Accessed: 2017-04-26.

[6] Florian Holzschuher and René Peinl. Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 195–204. ACM, 2013.

[7] Oleksii Kononenko, Olga Baysal, Reid Holmes, and Michael W. Godfrey. Mining modern repositories with elasticsearch. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 328–331, New York, NY, USA, 2014. ACM.

[8] Scikit learn developers. Scikit-learn. http://scikit-learn.org/stable/index.html, 2017. Accessed: 2017-04-26].

[9] Hannes Mühleisen and Christian Bizer. Web data commons-extracting structured data from two large web corpora. *LDOW*, 937:133–145, 2012.

[10] Neo4j. Neo4j, the world's leading graph database. https://www.neo4j.com. Accessed: 2017-04-26.

[11] University of Waikato. Weka. http://www.cs.waikato.ac.nz/ml/weka/, 2017. Accessed: 2017-04-26.

[12] Richard Boulton Olly Betts. Dnowball. http://snowball.tartarus.org/algorithms/dutch/stemmer.html, 2017.

[13] Michael E Porter. Location, competition, and economic development: Local clusters in a global economy. *Economic development quarterly*, 14(1):15–34, 2000.

[14] Tobias Preis, Helen Susannah Moat, and H Eugene Stanley. Quantifying trading behavior in financial markets using google trends. *Nature: scientific reports*, 2013.

[15] Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. Wikilinks: A large-scale cross-document coreference corpus labeled via links to wikipedia. *University of Massachusetts, Amherst, Tech. Rep. UM-CS-2012-015*, 2012.

[16] Jason R Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. Dirt cheap web-scale parallel text from the common crawl. In *ACL (1)*, pages 1374–1383, 2013.

[17] Lynn Wu and Erik Brynjolfsson. The future of prediction: How google searches foreshadow housing prices and sales. In *Economic analysis of the digital economy*, pages 89–118. University of Chicago Press, 2014.