The AOL Scandal
An Information Retrieval View

Wouter Roelofs

July 31, 2007

Contents

Chapter 1

The AOL Scandal

1.1 Introduction

Today, the World Wide Web is more important than ever. Practically everyone (in Western countries) uses the internet daily as a main source of information and communication. Google, the worlds largest search engine, is known and used all over the world and tries to 'organize the world's information and make it universally accessible and useful'. Given these facts, it is not hard to imagine that many people all over the world work in the branch of Information Retrieval.

Both the academic and the business world spend a lot of effort in this field. Researchers everywhere try to contribute to Information Retrieval by thinking up new and better ways of finding information and businesses everywhere try to implement these ideas and make a living out of it. The same goes for America OnLine (AOL), a global web services company that provides services for millions of people worldwide. It provides internet connections, instant messaging, e-mail service and their large social network to its customers. In some of those fields, like instant messaging, they are the largest in the U.S.

One of the major problems for the researchers on Information Retrieval is the lack of real life data. Search engines like Google keep this information secret for obvious reasons. This does give the academic world a headache since they can improve their research if they had this data. And going to work for Google is not an option for most of them. Three employees of AOL, however, knew the scientific community hardly had any live data. Therefore, they announced they would release 3 months of search data from AOL's customers on The First International Conference on Scalable Information Systems in Hong Kong in June 2006 [2]. Unfortunately for them, it did not work out to their advantage. Their action set the following events in motion:

August 4, 2006 AOL released a compressed text file on one of its websites containing twenty million search keywords for over 650,000 users over a 3-month period, intended for research purposes.

August 7, 2006 AOL pulled the file from public access, but not before it had been mirrored, P2P-shared and seeded via BitTorrent. News filtered down to the blogosphere[1] and popular tech sites such as Digg and Wired News.

[1]Blogosphere is the collective term encompassing all blogs as a community or social network

August 7-21, 2006 The following weeks the media continued exploring this hot issue, now named the AOL Scandal. The New York Times for example, tried to identify individuals by looking for names, addresses, social security numbers, etcetera in the data. They managed to track down several persons by cross referencing this data with phone books or other public records. A hefty ethics debate was the result.

August 21, 2006 AOL took the matter quite seriously which led to the resignation of AOL's CTO, Maureen Govern, the researcher who released the data, and his immediate supervisor, who reported to Govern.

September 7, 2006 I get my hands on a copy of the AOL search data, looked into the story behind it and decided to start this research.

September 22, 2006 A class action lawsuit is filed against AOL in the U.S. District Court for the Northern District of California. "The lawsuit accuses AOL of violating the Electronic Communications Privacy Act and of fraudulent and deceptive business practices, among other claims, and seeks at least $5,000 for every person whose search data was exposed", says Elinor Mills in her news blog. This adds up to a possible total amount of $3,250,000,000.

1.2 Research on the AOL Scandal

The growth of the internet and the way practically everyone uses it nowadays comes with a downside; our personal information can become freely available to everyone on the web. Therefore the discussion of privacy and especially digital privacy has become a pressing issue. Reason enough to have a look at this so-called AOL Scandal.

This AOL search data was released with the intention of stimulating academic researchers, so there must be something of value in it. There are many possible researches to be done with it. For example, it can be used to provide (relevant) feedback to the technique on finding useful search results. Or it can give information on who searchers judge the relevance of certain (types of) pages when searching on a certain query. Research can also be done on the composition of the average query in respect to what the searcher was actually looking for. Advertising companies might be interested in searching behaviour to address people personally.

After considering some of these options, the focus for this research was formulated. Keeping with my interest in Information Retrieval and specifically in Formal Concept Analysis (FCA), I decided to look at how the methods of FCA could be applied to (parts of) the AOL search data with the idea of uncovering some implicit links between people and their searching behaviour. Combining this with the hot issues of privacy and this so called AOL Scandal, I decided this was a feasible research for a bachelor thesis in Information Sciences.

1.2.1 Research Question

As with all good research, a research question was formulated for this research and divided into smaller questions which can (hopefully) be answered to answer the main question. The research question is as follows:

Is it possible to classify groups of people by their searching behaviour in an automated way using Formal Concept Analysis?

This question is quite specific and narrowed down, because a more general question would be too vague, leading to an unresearchable whole. The focus on classification is justifiable, because this fits quite nicely with the knowledge on and possible outcomes of FCA. On top of that, the possible affirmative outcome could be interpreted as an indication that further research on this subject could be wise. The results could also have a pragmatic value, which will be discussed in the final chapter of this thesis.

1.2.2 Sub Questions

In order to answer the main research question and to address some interesting side-issues, the following sub questions have been formulated, each with an explanation:

1. *Is the integrity of the data good enough to support valid outcomes?*
   One of the main ideas behind this research is that FCA can be applied to the AOL search data with a meaningful outcome because the data is fully representative for other real live search data. If the data set used for the research turns out to be compromised or manually adjusted, the whole outcome can be flawed and questioned. Therefore some meta-research must be done on the data as a whole.

2. *Is there an indication that the data supports FCA?*
   The research data may be one of the few real live sets on which FCA research get done, but it certainly is not the first. Previous research on the subject of Information Retrieval and more specifically FCA lead to some interesting results [6] that might also apply to this research. Therefore some statistical analysis will be done that will hopefully indicate that the data show some of the same features as other data sets used for FCA research.

3. *Does applying FCA to the data lead to relevant results?*
   As a third step in the process the actual FCA will be executed. The results of these experiments play the most important role in answering the main research question.

Two small side notes to reading the rest of this thesis:

1. The terms ‘the data’, ‘the search data’, ‘the research data’ and ‘the AOL data’ will all be used next to each other to denote the same thing, the AOL search data.

2. This thesis will obviously not contain any data that can actually be tracked back to individuals as this would be inappropriate and more importantly not ‘done’.

In order to answer these questions several methods will be used. Existing literature will be consulted on Information Retrieval techniques. Internet sources like renown news portals provide necessary information on the AOL Scandal itself. The practical part of the research will be done in a demo setting, using a copy of the AOL search data to do several experiments on. Because of the specific nature of the data, tools probably need to be made to exercise these experiments. Finally, there will be a presentation of the research done.

1.3 Outline of the Thesis

The thesis before you is the bachelor thesis of Wouter Roelofs, an Information Science student at the Radboud University of Nijmegen. The thesis is supervised by Dr. Franc Grootjen, who is specialized in the field of Information Retrieval. The thesis' structure is described in the rest of this section.

1.3.1 Parts

The thesis roughly consists out of five parts. The first is an introductory part that introduces the problem and describes the research plan. The second part will be a theoretical part, describing background information that support (the understanding of) the experiments. The third part contains the experiments themselves. This part has close links with the theoretical part. Their chapters can be mapped almost 1:1. The fourth part is the end of the research, which answers the question from the first. The fifth is an appendix part, this provides some additional information for completeness.

1.3.2 Chapters

Part 1 consists of just one chapter, the current one. It started with the introduction of the so-called AOL Scandal, giving a basic overview of events and the media circus around it. This is followed by the research plan, detailing what exactly will be researched. And it contains an outline of the thesis.

Part 2 has three chapters. Chapter 2 explains what search engines are and how they work. Some laws on computational linguistics are described in chapter 3. Chapter 4 provides you with an elaborate explanation of a data mining method called Formal Concept Analysis. This concludes the second part.

Part 3 is the practical part and also has three chapters. Chapter 5 covers some meta experiments on the data to ensure its usefulness. Chapter 6 links back to chapter 3 and tries to apply the linguistic laws to the research data. Chapter 7 then brings the theory from chapter 4 in practise in the experiments that will hopefully answer the main question.

The final real part of the thesis, part 4 will again have just one chapter. Chapter 8 is a typical conclusions chapter, answering the research questions. It also contains some thoughts on what continuing research could be done in the future.

For completeness, appendix A describes the way the research data was transformed into a more processable form.

Figure 1.1 sums up the structure of the thesis. To get a quick impression of the research, read the introduction and the conclusion chapters, chapters 1 and 8. Readers that are already familiar with the theory behind search engines, computational linguistics and FCA can skip the theoretical part and continue reading in chapter 5. People interested in the whole thing, simply continue in chapter 2.

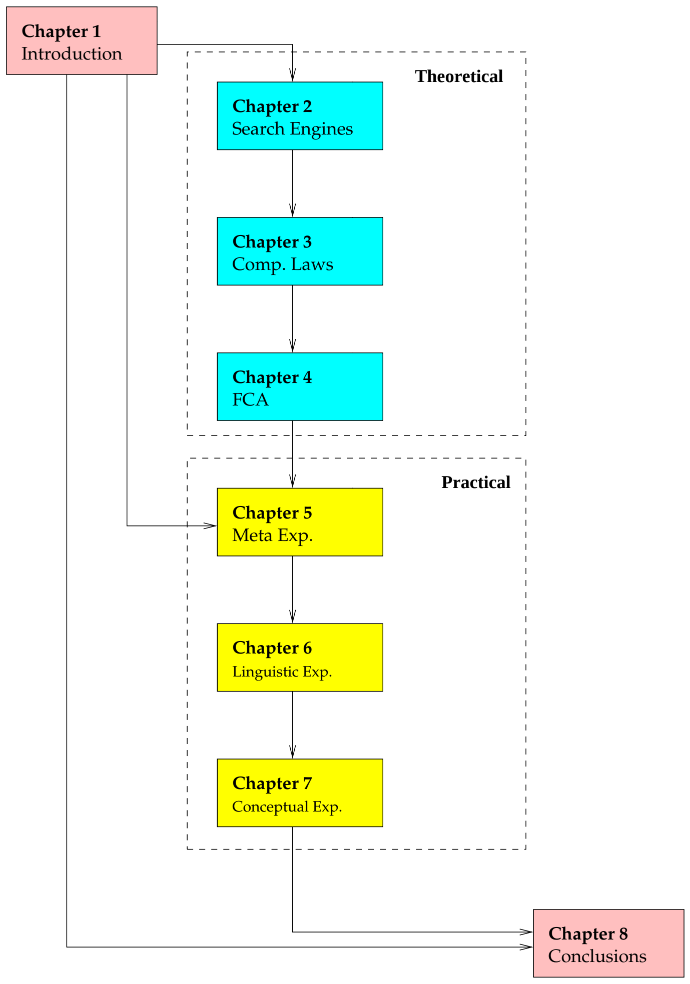


Figure 1.1: Outline of the Thesis

Chapter 2

Search Engines

2.1 Introduction

The search engine used by America OnLine is not actually their own search engine, it is a portal to Google. Buying existing search technique is a smart thing to do when you are not in that line of business yourself. And Google *is* the worlds largest search engine after all.

So the research data, the AOL search data, is actually mostly Google's data; just the notion of which person is searching is what AOL keeps track of. This is because those persons are logged in using their AOL account. The rest of the searching process gets done by Google. Of course we need to know what exactly we are looking at. Therefore this chapter will provide background information on how search engines work.

2.2 History Of Searching

Searching for information is something human kind has done since the beginning. We are always in a conquest for finding new information, because we believe knowledge can make us stronger[1]. In fact, computers were made to do information searching. Searching is a very structured but time consuming task, therefore we made a machine that could do it for us.

With the coming of the internet, the accessibility to information has been multiplied by a large factor. But with more information comes more trouble finding exactly what you want. And so the need for an internet search engine was born. The very first tool for searching the internet dates back to 1990, but this was a quite primitive system only capable of searching in filenames. The first real web search engine, called Wandex, was developed in 1993 and could search inside documents. The well-known search engine Google started in 2000 and added a couple of smart features to enhance the search results. Nevertheless, Google still operates in more or less the same way as any other search engine.

2.3 Why they work

When designing an automatized search engine you immediately encounter a problem: A computer cannot think like humans, a computer cannot reason, it can simply follow orders.

[1]The author is aware of the fact that information is not equivalent to knowledge, but this is a whole other discussion which lies outside the scope of this research

And people do not always know exactly what they are looking for either. After all, if you knew exactly what document you were looking for, you would not have to look for it! That is why you need to think up a way to communicate with the computer to 'explain' what it is you are looking for. And on the other side, the computer needs some sort of representation of the knowledge at hand and a way to match both.

This is why almost all automated retrieval systems use a variant of the *term-based* disclosure system. The main principle in this approach is that terms hold a descriptive (conceptual) value. These systems try to characterise documents using word sets. The words are normally taken from the document in an automated fashion. Essentially these systems assume that:

> If a certain word occurs in a document, that document is *about* that word.

Note that *word order* plays no role in these models. Consider the famous example of a document called 'The Hillary Clinton Bill Proposal' which contains the words 'Bill' and 'Clinton'. It illustrates the importance of word order, which is lost in this approach. Matching in this model involves comparing word occurrences in query and characterisations, which can be done in numerous ways. All term-based models rely on the following hypothesis:

> *If a document and the query have a word in common, the document is likely to be relevant.*

And the funny thing about this is that this seems to work. Up to today, most efforts done by linguistic researchers on using word syntactics or semantics to improve search systems have resulted in systems that are not much better than these simple systems of counting which words occur in a certain document and which do not.

2.4 The Structure

A typical web search engine consists of three parts: Web Crawling, Indexing and Searching. The internet is a worldwide network between computers, therefore no-one has all the data on the web stored internally. This makes it a bit harder to 'know' what information you have got. If you compare it to having a number of files on your computer. If you would like to search in them, you can always simply access them, you know if they were changed, etcetera. The internet is not like that, therefore there is the technique called Web Crawling, this part handles the locating of web pages. When you found those web pages, you would probably like to know what is inside them, so you can give a more accurate search result to the searcher. This is done by the second part, the Indexing. And finally there is the part where a person actually sits behind its desk and would like to find a web page containing the information he was hoping for, Searching.

2.5 Web Crawling

Staying with the metaphor of the internet being a web of computers, web crawlers are often called spiders. They are little programs that 'walk' the web looking for web pages and doing some additional work on the way, such as keeping statistics. But how do you find (almost) every page on the web? Web crawlers do this by simply following the links on each page to a next page. These links were made by the designers of the web pages and together they make a complete web of pages. Logically search engines like Google also offer people that just made a new website the opportunity to tell them where it is. This gives web crawlers a new page to start crawling.

There are two different styles of crawling, depth-first and breadth-first. In the first case when a spider comes across a link, it immediately follows it and goes back following other links when it reaches a dead end. If you would draw a schematic picture of this situation, putting the first page at the top and every page it links to beneath it, you will see that by this way of walking through the pages the spider goes deep first, hence the name depth-first. In the second case, the spider queues up all the links on a page and then follows them in that order before moving to the next level of deepness. This time the spider goes wide first, then goes deep. See figure 2.1 for an example. The breadth-first is the more popular choice because it results in higher quality pages.

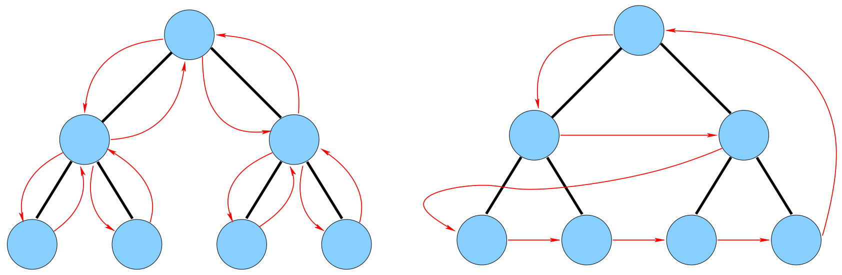


Figure 2.1: Depth-first versus Breadth-first

Please note that this is a simplified view on how a web crawler works. In practice many techniques have been thought up over the years to enhance the work of web crawlers. Among these techniques are: clever selecting of pages, avoiding getting into endless loops, path-ascending crawling, focused crawling and revisiting. A more detailed overview of crawlers and crawling techniques unfortunately lies beyond the scope of this research.

2.6 Indexing

When a spider reaches a web page, it needs to do something with it so that the search engine know this page exists and what it is about. After all, we want to provide the searchers with the best fitting answers. Therefore, we index. Obviously we will not save the pages as a whole, because every search engine would then hold a copy of the internet on their computers. That is why people needed to come up with a better idea on how to keep track of the contents of a page.

As we saw before, the system we use is based on counting the occurrence of words and that is exactly what is done when indexing. In the case of Google, the indexing is done by the same piece of software that does the crawling, the spider. So it has crawled to a certain web page and now it simply goes over the contents of the (source of) the page and tallies the words and their occurrences. Modern indexing software does a bit more than simple tallying work, but it still is the main idea behind and strength of web search engines.

A couple of important properties of indexing are: the size of the index, the speed of looking things up, the way the indexed data is stored and maintenance of the index over time. For many of these issues clever solutions have been thought up, but those are not in the scope of this research.

What still needs to be mentioned however is the way the index is kept, because this affects the way you can search in it later on. Many search engines incorporate an inverted index. This means that, instead of keeping a list of documents and per document their words, it keeps a list of all words used and per word which document it occurs in. See table 2.1 for an example.

Word	*Documents*
the	Document 1, Document 3, Document 4, Document 5
cow	Document 1, Document 2, Document 3, Document 4
says	Document 1, Document 5
moo	Document 1, Document 7

Table 2.1: An inverted index

2.7 Searching

Finally we come to the actual searching. Now we have an inverted index of a set of documents on the internet and there is a certain information need from a searcher. We have already seen that if a document and the query have a word in common, the document is likely to be relevant. And we have seen that the technique most search engines use is a relatively simple one, since it only deals with word occurrences in documents instead of actual language semantics. It does not come as a surprise then that the search part of search engines is term-based too.

The searcher formulates his information need in a so-called *query* which seems to be a highly unstructured and ambiguous construct. For example, when I am searching for a web page on magical squares, my query could simply be 'magical squares'. Both words can mean many things, when I say 'magical' you could think about magicians, illusionists or even simply amazing things. 'Squares' could indicate a square in a city, like the famous Trafalgar Square. Nevertheless a search engine will come up with exactly the web pages I would like to see, about the mathematical phenomenon. So how does an automated system with no actual knowledge do this. We will now look at two ways, using the 'classical' weight tf-idf and using the technique that contributed to making Google so incredibly popular: PageRank.

2.7.1 Tf-idf

The tfidf weight (term frequencyinverse document frequency) is a weight often used in Information Retrieval and text mining [11]. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

The *term frequency* in the given document is simply the number of times a given term appears in that document. This count is usually normalized to prevent a bias toward longer documents (which may have a higher term frequency regardless of the actual importance of that term in the document) to give a measure of the importance of the term t_i within the particular document.

$$\mathit{tf}_i = \frac{n_i}{\sum_k n_k}$$

where n_i is the number of occurrences of the considered term, and the denominator is the number of occurrences of all terms.

The *inverse document frequency* is a measure of the general importance of the term (obtained by dividing the number of all documents by the number of documents containing the term, and then taking the logarithm of that quotient).

$$\mathit{idf}_i = \log \frac{|D|}{|\{d : d \ni t_i\}|}$$

with

- $|D|$: total number of documents in the corpus
- $|\{d : d \ni t_i\}|$: number of documents where the term t_i appears (that is $n_i \neq 0$).

Then

$$\mathit{tfidf} = \mathit{tf} \cdot \mathit{idf}$$

A high weight in tfidf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tends to filter out common terms.

And so search engines calculate the tf-idf weight for a given query and displays the documents that have the highest rating, ordered by that rating of course.

2.7.2 PageRank

Using the tf-idf weight works surprisingly well, high results have been made by it, especially after improving it with some smart tricks. If you are searching for a web page on magical squares however and a certain web page is a phony web simply containing the words 'magical' and 'squares' a lot of times, it would result in a high tf-idf score, thus would be assumed to be very fitting to your request. But you probably do not want this page, therefore Larry Page, Co-Founder & President of Products of Google Inc, developed PageRank.

PageRank is a link analysis algorithm that assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. A simple description of PageRank by Google: 'PageRank relies on the uniquely democratic nature of the web by using its vast link structure as an indicator of an individual page's value. In essence, Google interprets a link from page A to page B as a vote, by page A, for page B. But, Google looks at more than the sheer volume of votes, or links a page receives; it also analyzes the page that casts the vote. Votes cast by pages that are themselves "important" weigh more heavily and help to make other pages "important"'.

PageRank is a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for any-size collection of documents. It is assumed in several research papers that the distribution is evenly divided between all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called ”iterations”, through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

The (simplified) PageRank algorithm uses the following equation:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

where $p_1, p_2, ..., p_N$ are the pages under consideration, $M(p_i)$ is the set of pages that link to p_i, $L(p_j)$ is the number of outbound links on page p_j, and N is the total number of pages.

Using this PageRank algorithm Google is able to get even better fitting results for search queries than the original tf-idf approach. Going into these techniques in more extend goes beyond the scope of the research.

2.8 The AOL Search Data

The AOL search data is data of the third part, so it is data of the actual search process done by humans. It contains queries by users and the results that the AOL search portal gave (through Google). The really interesting part is that the queries are linked to users, although anonymized. This is information you would normally not have, since you do not have to login to use Google. Google could of course log your ip-address, but that would not give much assurance as people might have a dynamic ip or go to school and search there. And it could be that a lot of people search from the same ip-address. This makes the AOL search data interesting to experiment with.

We conclude this chapter with the following figure that sums it up.

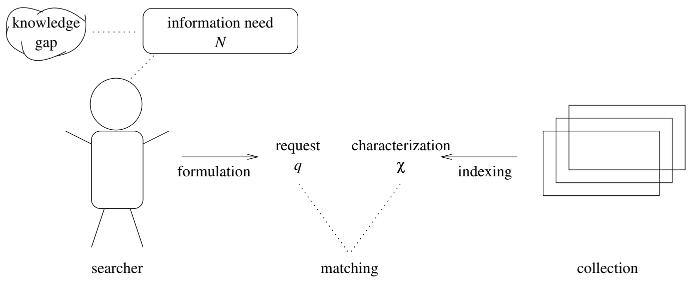


Figure 2.2: The Information Retrieval Paradigm

A searcher had a knowledge gap, which leads to an information need N. The searcher formulates his information need in a request q. From the other side, there is a collection of documents. By indexing the characterization χ of the documents. Finally, these two sides come together in the processing of matching, so the searcher can find the documents that fill his knowledge gap.

Chapter 3

Laws in Computational Linguistics

3.1 Introduction

In order to fully understand the reason behind the experiments described in chapter 6 we must first look at language and, more specifically, at some unwritten laws that appear in language. The laws we are talking about are called Zipf's law and Heaps' law. Both Zipf and Heaps found logical patterns in (natural) language and formulated these in mathematical laws.

This chapter will present these laws, their characteristics and properties and will hopefully provide enough background information for chapter 6.

3.2 Zipf's Law

There are many models that, given the probability of certain events, try to describe the probability of a sequence of those events. The most well-known models that try to connect these probabilities are *Zipf's Law* [16] and the *Mandelbrot distribution* [10].

Zipf's Law states that, in many practical situations, if the r-th most probable event has probability p, then $p \cdot r$ is (almost) equal for all events. The Mandelbrot distribution claims this for the expression $p \cdot (c+r)^\theta$ for some parameters c and θ. In case of $c = 0$, the distribution is also referred to as the generalized Zipf's Law. Some authors motivate the validity of these laws from physical phenomena, see for example [15] for Zipf's Law in the context of cities. But it is also possible to derive Zipf's law from a simple statistical model [9].

Probably the best known application of Zipf's law is from linguistics: Zipf's Law can used to describe word frequencies in natural language input. For example Tolkien's *The Lord of The Rings* [12] produces figure 3.1 when we plot word frequencies against word ranks on a double log scale.

The simplest case of Zipf's law is a "$1/f$ function". Given a set of Zipfian distributed frequencies, sorted from most common to least common, the second most common frequency will occur 1/2 as often as the first. The third most common frequency will occur 1/3 as often as the first. The nth most common frequency will occur $1/n$ as often as the first. And this is exactly what we see in figure 3.1. However, this cannot hold precisely true, because items must occur an integer number of times: there cannot be 2.5 occurrences of a word. This is the reason why we stated 'is (almost) equal' before. Nevertheless, over fairly wide ranges, and to a fairly good approximation, many natural phenomena obey Zipf's Law.

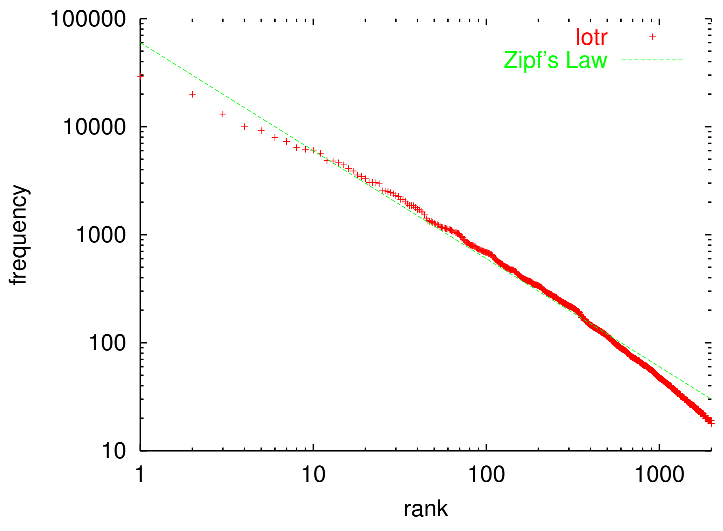


Figure 3.1: Word frequencies from The Lord of the Rings

3.2.1 Mathematical

For completeness sake we conclude this section by showing the mathematical formulation of Zipf's Law. It can be formulated as:

$$f(k; s, N) = \frac{1/k^s}{\sum_{n=1}^{N} 1/n^s}$$

where N is the number of elements, k is their rank, and s is the exponent characterizing the distribution. In the example of the frequency of words in the English language, N is the number of words in the English language and, if we use the classic version of Zipf's law, the exponent s is 1.

3.3 Heaps' Law

Another (experimental) law of nature is Heaps' Law [7], which describes the growth in the number of unique elements (also referred as the number of records), when elements are drawn from some distribution.

Heaps' Law states that this number will grow according to αx^β for some application dependent constants α and β, where $0 < \beta < 1$. In the case of word occurrences in natural language, Heaps' Law predicts the vocabulary size from the size of a text. In figure 3.2 this relationship is presented for *The Lord of the Rings* together with a fit[1] of Heaps' law for $\alpha = 36.7$ and $\beta = 0.46$.

A lot of (linguistic) research has been done to statistically describe word distributions. Only the fact that languages are *closed* (having a finite number of words) or *open* has been the source of a lot of discussion. In [8] a good historic overview is given. Furthermore it contains an argumented claim that languages are *open*.

[1]Fit has asymptotic standard error of 0.75% for α and 0.13% for β

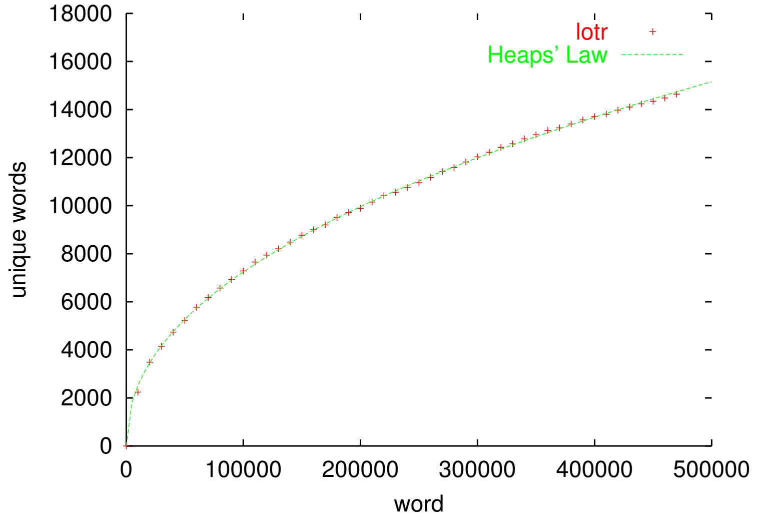


Figure 3.2: Vocabulary growth of The Lord of the Rings

3.3.1 Mathematical

Like with Zipf's Law, we shall quickly go over the mathematical definition of Heaps' Law. It can be formulated as:

$$V_R(n) = \alpha n^\beta$$

where V_R is the subset of the vocabulary V represented by the instance text of size n. α and β are free parameters determined empirically. With English text corpora, typically α is between 10 and 100, and β is between 0.4 and 0.6. As we saw previously, this is the case when looking at *The Lord of the Rings*.

3.4 Link with the AOL Scandal

What we have not discussed in this chapter yet is what these laws of language have to do with the AOL Scandal. Our main assumption in this research was: The words a person looks for tell us something about that person. So if the AOL Search Data would unconsciously follow the same patterns as natural language, then these two laws might apply to them too. This leaves us the opportunity to use some of the knowledge we have on natural language and use it to our advantage on the research, like that of stop words.

3.4.1 Stop Words

Stop words are words that are generally seen as 'less meaningful words'. The term was introduced by Hans Peter Luhn, one of the pioneers in information retrieval, who used the concept in his design and implementation of KWIC indexing programs. Typically, stop words are adverbs and articles. Some examples are 'the', 'a', 'in', 'of'. They are said to add less information to linguistic constructions than for instance nouns. Nevertheless they have a purpose. The purpose of stop words is twofold:

1. *Redundancy*
   Stop words have a redundancy function. Especially in spoken language where people might not hear each other very well, stop words limit the amount of possible errors. For example, when person A says to person B 'it is in the pocket of my coat', the words 'in' and 'of' makes it illogical for B to think A said 'it is in the pocket of my boat', since boats do not have pockets. Therefore these little added words can add a lot to the understanding. Note that this does not apply to written language.

2. *Construction*
   Stop words also have a purpose in constructions: 'in the closet' means something completely different than 'on the closet'. And 'president' means something different than 'the president'. These stop words really add extra information in these cases. However, if you would disregard constructions, their added meaning would decrease again.

This research uses a word model in which all words are used separately. In such a case, words like 'of' and 'in' are treated equally to nouns. This could mean that leaving in stop words is a somewhat bad idea. We will come back to this issue in chapter 6.

Chapter 4

Formal Concept Analysis

4.1 Introduction

In previous chapters we have looked at some theoretical basics of subjects from the Information Retrieval world. Chapter 2 has shown how search engines work and chapter 3 dealt with laws in computational linguistics. This final theoretical chapter will give a basic overview of Formal Concept Analysis. After this chapter we will have enough knowledge to conduct our experiments in hope of answering the research question.

Formal Concept Analysis, from here on abbreviated to FCA, is a method of data analysis that was introduced in 1982 by R. Wille [14] and applied in many quite different realms like psychology, sociology, anthropology, medicine, biology, linguistics, computer sciences, mathematics and industrial engineering. Worldwide there are dozens of academical institutes all doing research on the subject. But what is FCA and how can it be of use to us?

4.2 Concepts

FCA deals with concept. What exactly is a concept? According to Merriam-Webster a concept is:

```
Main Entry: concept
Pronunciation: 'kn-"sept
Function: noun
Etymology: Latin conceptum, neuter of conceptus, past participle
  of concipere to conceive -- more at CONCEIVE
1 : something conceived in the mind : THOUGHT, NOTION
2 : an abstract or generic idea generalized from particular
  instances
```

This is a rather vague description, but it does give us an idea of what a concept is. Merriam-Webster talks about 'conceived in the mind' so it has to do with human thinking. The description also says 'abstract or generic idea' so it has to do with abstracting from specific examples and finding a higher level structure.

We will try to understand what a concept is with the following example. When you read the word 'chair' somewhere, you probably know what is meant by it. Undoubtedly, you have seen chairs because everyone uses them in daily life. Perhaps your neighbours have other chairs than you, but still you recognize them as being chairs. We can also explain what a chair is by looking at what makes them chairs, their properties. You could say that a chair is something with four legs, a back and is something to sit on. These are both ways of looking at the *concept* chair.

4.3 Formal Concepts

It is only a short step from concepts to formal concepts; they have a lot in common. Nevertheless, a formal concept is, as the name predicts, formally defined thus (hopefully) easier to grasp. To see what a formal concept is we will look at a returning example. It is an example of the planets in our solar system. In our solar system, or *context* to keep it abstract, there are nine planets: Mars, Neptune, Uranus, Saturn, Jupiter, Earth, Pluto, Venus and Mercury. Some of these planets have moons circling around them, others do not. They all have a distance to the sun, some are nearer than others. And the planets differ in size too, some are big, others small. We abbreviate these properties and put the planets and these properties in a table, figure 4.1.

Planets	no moon	moon	near	far	small	medium	large
mars		×	×		×		
neptune		×		×		×	
uranus		×		×		×	
saturn		×		×			×
jupiter		×		×			×
earth		×	×		×		
pluto		×		×	×		
venus	×		×		×		
mercury	×		×		×		

Figure 4.1: The planets context

In this context the planets Mars and Earth share the properties moon, near and small. They are the only ones that share exactly these properties and they share no more properties. The combination of these planets and these properties is therefore called a formal concept. Formal concepts can prove to be quite intuitive too. The formal concept we just found can be interpreted as 'earth-like planets', planets that can possibly harbor life.

4.4 Formal Definition

Now to make things really formal. We denote the collection of objects as $\mathcal{O}$, the individual members of this collection we write o_1, o_2 etc, while we write subsets as O_1, O_2 etc. The same goes for the collection of attributes, but with $\mathcal{A}$, a_1 and A_1. In this definition we use the words 'objects' and 'attributes', because it fits our planet example. You can abstract from these specific terms and talk about any given collections in which the members have a binary relation to each other. The binary relation is defined as $\sim \subseteq \mathcal{O} \times \mathcal{A}$ and:

$$o \sim A \iff \underset{a \in A}{\forall} o \sim a$$

$$a \sim O \iff \underset{o \in O}{\forall} a \sim o$$

For the relation $\sim$ we define the right polar function **rightPolar**: $\mathcal{P}(\mathcal{O}) \rightarrow \mathcal{P}(\mathcal{A})$ as follows:

$$\textbf{rightPolar}(O) = \{a \in \mathcal{A} | a \sim O\}$$

And we define the left polar function **leftPolar**: $\mathcal{P}(\mathcal{A}) \rightarrow \mathcal{P}(\mathcal{O})$ as:

$$\textbf{leftPolar}(A) = \{o \in \mathcal{O} | o \sim A\}$$

The **rightPolar** and **leftPolar** functions are in general not each others inverse. When they are, a special situation occurs: a formal concept is a pair $(O, A) \in \mathcal{P}(\mathcal{O}) \times \mathcal{P}(\mathcal{A})$ with:

$$\textbf{rightPolar}(O) = A$$

$$\textbf{leftPolar}(A) = O$$

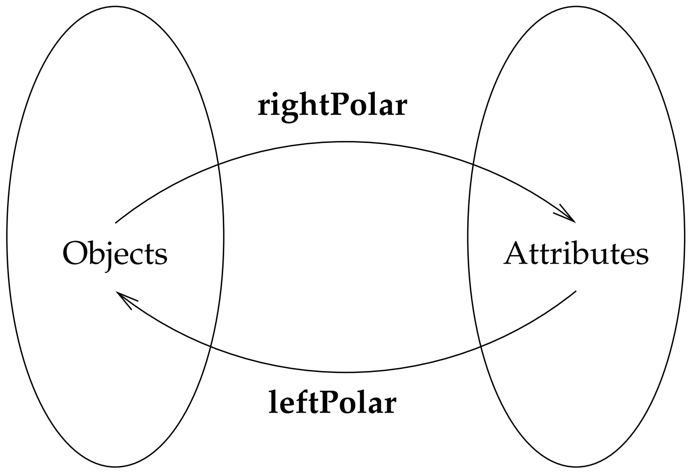


Figure 4.2: The dualistic view with right and left polar function

4.5 Concept Lattices

These concepts can be partially ordered by inclusion: if (O_i, A_i) and (O_j, A_j) are concepts, we define a partial order $\leq$ by saying that $(O_i, A_i) \leq (O_j, A_j)$ whenever $O_i \subseteq O_j$. Equivalently, $(O_i, A_i) \leq (O_j, A_j)$ whenever $A_j \subseteq A_i$. Every pair of concepts in this partial order has a unique greatest lower bound (meet) and a unique smallest upper bound (join), so this partial order satisfies the axioms defining a lattice.

The greatest lower bound of (O_i, A_i) and (O_j, A_j) is the concept with objects $O_i \bigcap O_j$; it has as its attributes the union of A_i, A_j, and any additional attributes held by all objects in

$O_i \bigcap O_j$. The smallest upper bound of (O_i, A_i) and (O_j, A_j) is the concept with attributes $A_i \bigcap A_j$; it has as its objects the union of O_i, O_j, and any additional objects that have all attributes in $A_i \bigcap A_j$.

In case of our example, a lattice of all concepts would look like figure 4.3. Each concept in the lattice is only shown with its attributes. This has two reasons:

1. Adding the objects would make the figure unreadable.
2. The way it is now, you can clearly see the ordering of the concepts over the different levels. For instance, the concept ({mars, earth},{moon, near, small}) we saw earlier is a 'larger' concept than ({mars, earth, venus, mercury},{small, near}) and ({mars, earth, pluto},{moon, small}), because those only have two of the three attributes. And of course the other way around 'smaller' concepts have more objects than their direct larger ones.

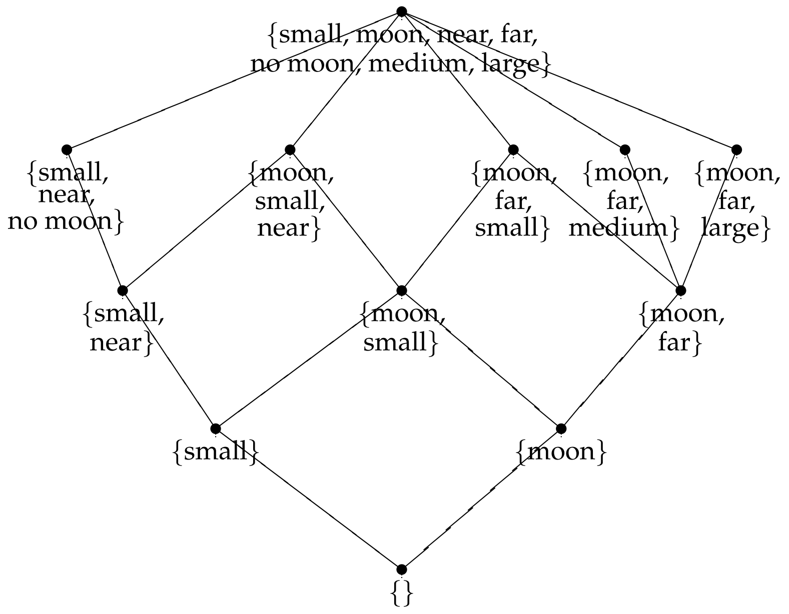


Figure 4.3: Planets concept lattice

4.6 Finding Formal Concepts

There are several ways of finding (formal) concepts in a given context. We will look at three examples: the simple, intuitive way that has a very high complexity, a somewhat intuitive way that requires a lot less work but still is quite energy consuming and finally a clever method that can really minimize the time needed to compute the concepts.

4.6.1 Basic

The first algorithm is the simplest and most straightforward. Simply generate all possible subsets O from $\mathcal{O}$ and A from $\mathcal{A}$ and check whether (O, A) is a concept. You can check

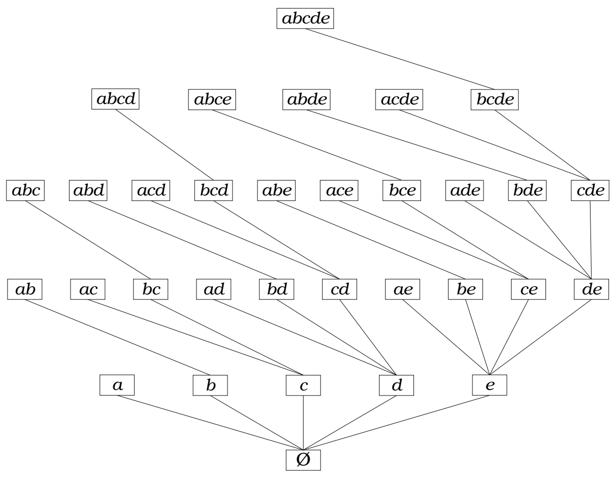


Figure 4.4: order for $E = \{a, b, c, d, e\}$

whether (O, A) is a concept by applying the **rightPolar** function to O and see if you get A and the **leftPolar** function to A and see if you get O. This method obviously is very time consuming. Assuming n objects in O and m attributes in A, it has a complexity of 2^{n+m}. The number of concepts that can be found is bounded by $\min(2^n, 2^m)$ [4].

4.6.2 Meet and Join

In practice, the number of concepts will be far less. As a consequence, it will be profitable to bound the complexity of the generation algorithm to the actual number of concepts. And that is what this second method does, it uses the knowledge of lattices. We know the concepts are ordered, so we can formally derive a higher level concept from lower level concepts. Therefore, we can generate the entire lattice by starting with the base concepts and calculating the closure with respect to the join-operator [4]. And this is what the second method does. As you can imagine this method is a lot more efficient since it no longer checks all possible subsets of the objects and attributes collections. However, it is still rather complex to calculate larger concepts from smaller ones. And since the number of concepts has a size of $O(n^3)$ (where n is the number of attributes) [6], this method, especially for large collections, is still not very fast.

4.6.3 Ganter

The third and last algorithm we will discuss here is the algorithm developed by Bernhard Ganter [3]. The Ganter algorithm solves the major problem there is with the first method. The major flaw in brute-force calculation of all concepts is that it simply does way too much unnecessary work. Just a fraction of all possible subsets of the object and attributes collections need to be evaluated. And Ganter found a way to limit the amount of work.

He discovered that if you order all possible subsets of objects in a certain way, the Ganter algorithm effectively goes depth-first through a binary tree (figure 4.4).

He then mathematically proved that that *if* a node in the order tree is a concept *then* the branch will not yield any more concepts, [3]. Because of this Ganter's algorithm can stop searching for concepts in certain branches and is faster than a brute force implementation, which needs to check each possible subset of objects.

In 2006 Mark Blokpoel [1] did research on a parallel implementation of the Ganter algorithm in Java. His implementation offers a clean and simple way of doing FCA. It supports multiple forms of inputting contexts and outputting the resulting concepts. It does not however keep any notion of relations between concepts. A short excerpt from the output of Blokpoel's program after running the planets example:

```
<concept>
  <yobject>mars</yobject>
  <yobject>earth</yobject>
  <yattribute>moon</yattribute>
  <yattribute>near</yattribute>
  <yattribute>small</yattribute>
</concept>
```

Chapter 5

Meta Experiments

5.1 Introduction

The first step in a research should be to check the research data used for consistency. If the data proves to be tempered with, made up or simply out of balance, the whole research can be questioned. Therefore some meta-research must be done on the data. In this chapter the data will be examined, compared to the claims AOL itself makes about it and checked for any anomalies

5.2 General Statistics

The AOL search data has been spread in a plain text format, zipped into an archive file. The original data file as published by AOL could be found at their research site under '500kUserQueriesSampledOver3Months'. Obviously, this file has been removed from their site. If you are looking for the data however, you might still be able to find it mirrored somewhere. Such a copy is used for this research.

The archive file is called `AOL-user-ct-collection.rar` and is approximately 321.079 kB in size. Inside we find a directory named `AOL-user-ct-collection` in which are 11 plain text files: `U500k_README.txt` and ten files named `user-ct-test-collection-{01 to 10}.txt`. The first file explains what the collection is and what data is in it. The following statistical information is from that file:

```
Basic Collection Statistics
Dates:
  01 March, 2006 - 31 May, 2006

Normalized queries:
  36,389,567 lines of data
  21,011,340 instances of new queries (w/ or w/o click-through)
   7,887,022 requests for "next page" of results
  19,442,629 user click-through events
  16,946,938 queries w/o user click-through
  10,154,742 unique (normalized) queries
     657,426 unique user ID's
```

As you can see this is all statistical information on meta-level. statistical information on the actual data will be discussed in chapter 6.

To make the data a bit easier to handle, it has been read into a MySQL database. Now the power of querying the data can be exploited using the SQL syntax, which makes it easier to gather or check simple information like how many queries are there. Basic knowledge about SQL is recommended for further reading. If you are unfamiliar with SQL, you can check the Wikipedia page on it. For detailed information on the database structure and the method used to insert the AOL data into the database, see appendix A.

5.3 Checking

A first step is to check whether these general statistics as claimed by AOL are correct. Since we have the SQL database and therefore the power of SQL, this can be done with ease.

5.3.1 Lines of Data

Checking the total number of lines is quite easy.

```
mysql> SELECT COUNT(*)
    -> FROM data;
+----------+
| COUNT(*) |
+----------+
| 36389567 |
+----------+
1 row in set (0.08 sec)
```

AOL said there were `36,389,567 lines of data` so this is correct.

5.3.2 Instances of New Queries

Instances of new queries is a bit trickier. The AOL read-me file describes the different situations you can have while searching on the internet and how these are stored in the AOL data. One of the things it describes is how 'next page of results' are saved. Whenever a user did a query, but did not find what he was looking for, he goes to the next page of results. This appears in the search data as two different lines, containing the same query, but with a later time stamp. As an opposite there are of course the queries that did not result in a next page requests and as you can imagine, these two numbers should add up to the total number of queries done. Let's begin with looking at the new instances of queries, so the ones that are not a next page request.

```
mysql> SELECT anonid, query, COUNT(*)
    -> FROM data
    -> GROUP BY anonid, query
    -> HAVING COUNT(*) = 1;
21008404 rows in set (98 min 20.52 sec)
```

As you can see, AOL claimed there to be `21,011,340 instances of new queries` so there is a slight discrepancy here. This is probably caused by similar queries done by the same user at later times (which are in fact new queries, but this rather rude SQL query does not take that into consideration). Since it is only off by 0,0014% we assume the number is correct.

5.3.3 Next Page Requests

Now for the counterpart. We repeat the query, but this time take the cases where there are multiple similar queries done. These obviously are the next page requests.

```
mysql> SELECT anonid, query, COUNT(*)
    -> FROM data
    -> GROUP BY anonid, query
    -> HAVING COUNT(*) > 1;
15381163 rows in set (65 min 15.01 sec)
```

AOL said there were `7,887,022 requests for "next page" of results` and the query shows a number of 15,381,163 so there is again a mismatch. But this time it is rather large. With a factor of almost 2 AOL must use an other definition of next page requests. Fact is that the numbers found here are internally consistent; 21,008,404 (new instances) + 15,381,163 (next page requests) = 36,389,567 (total). While AOL's number are not; 21,011,340 + 7,887,022 = 28,898,362. Unfortunately, there is no one to ask at AOL about this difference, but we assume that AOL messed up their definitions, as their numbers are not internally consistent.

5.3.4 Click-through Events

Click-through events are those queries on which a user has acted by clicking on of the links in the results. In the data these are indicated by having a value in the last two fields, namely the ItemRank and ClickUrl fields. Now we can query the database in the following way.

```
mysql> SELECT COUNT(*)
    -> FROM data
    -> WHERE clickurl != '';
+----------+
| COUNT(*) |
+----------+
| 19442629 |
+----------+
1 row in set (2 min 29.52 sec)
```

As we can see this number matches the 19,442,629 that U500k_README.txt says.

5.3.5 Queries without Click-through

This is an easy one, as it is the complement of the click-through events in respect to the total number of lines. 36,389,567 (total) - 19,442,629 (click-throughs) = 16,946,938 (not click-throughs).

For completeness' sake however, we will run a check-up query.

```
mysql> SELECT COUNT(*)
    -> FROM data
    -> WHERE clickurl = '';
+----------+
| COUNT(*) |
+----------+
| 16946938 |
+----------+
1 row in set (2 min 22.47 sec)
```

Now we see that we were correct and this number is indeed the complement.

5.3.6 Unique Queries

Calculating the number of unique queries is not really hard to do. We just need to count the distinct queries.

```
mysql> SELECT COUNT(DISTINCT query)
    -> FROM data;
+-----------------------+
| COUNT(DISTINCT query) |
+-----------------------+
|              10154429 |
+-----------------------+
1 row in set (25 min 16.63 sec)
```

`10,154,742 unique (normalized) queries` is what the readme file said. There appears to be a small gap of 313 queries. Since this is 0,003% of the total amount it probably has to do with the '(normalized)' part. Since there is no notion on how and what was normalized and possibly unnormalized when making the text files, we assume their claim is correct.

5.3.7 Unique ID's

Checking unique id's is almost as easy as checking the lines of data, we just need to use a 'DISTINCT' keyword.

```
mysql> SELECT COUNT(DISTINCT anonid)
    -> FROM data;
+------------------------+
| COUNT(DISTINCT anonid) |
+------------------------+
|                 657427 |
+------------------------+
1 row in set (1 min 42.19 sec)
```

Here we find another inconsistency. The claim was that there were `657,426 unique user ID's`, however we found one more. When looking at this occurrence, we remember that when we originally read in the database, we deleted the ten 'header' lines that were in the ten plain text files. These all had 'ClickURL' as clicked url and therefore we threw them away. However, when we check for anonid '0' once more, we notice that there is still one line left with this id.

```
mysql> SELECT *
    -> FROM data
    -> WHERE anonid='0';
+--------+-----------+---------------------+----------+----------------------+
| anonid | query     | querytime           | itemrank | clickurl             |
+--------+-----------+---------------------+----------+----------------------+
|      0 | match.com | 2006-03-31 06:55:53 |        2 | http://www.match.com |
+--------+-----------+---------------------+----------+----------------------+
1 row in set (2 min 5.39 sec)
```

Apparently this is a valid query, since it was included in the total line count of 36,389,567 queries, so we assume that for this particular query something went wrong when determining the anonid and they simply left it at '0'. Therefore we assume that the number of unique user id's is correct.

5.4 Queries per Day

As we search for anomalies on first sight in the AOL search data, we should also check the distribution of the queries. In order to do so we perform another simple query to our database, this time writing the result to a file.

```
mysql> SELECT DATE(querytime), COUNT(*)
    -> FROM data
    -> GROUP BY 1
    -> ORDER BY 1
    -> INTO OUTFILE 'queriesperday.dat';
Query OK, 92 rows affected (2 min 34.06 sec)
```

Then, after plotting 'queriesperday.dat' we get the following figure.

5.4.1 Rythm

Looking at figure 5.1 we notice that there seem to be a weekly rhythm in the amount of queries per day. There is one clear peak which returns every seven days. With the first peak occurring on 05-03-2006, which was a Sunday, we conclude that people seem to search more on Sundays, probably because it's their day off.

We also see that there seems to be a rhythm in general, differences from week to week. This could have to do with national holidays or disasters. There appear to have been no disasters around 13-04-2006, maybe it had something to do with Easter.

5.4.2 Missing Day

There is one other thing to mention though, the number of queries on 17-05-2006. Just 3,349 queries were done that day, in contrast to the average of 395,539 queries per day (or 399,849 without that day). This is a difference of a factor 118! This could indicate three things:

1. People collectively decided not to use the internet the 17th of May
2. AOL intentionally left out a lot of results of this day
3. There must have been a server-down at AOL

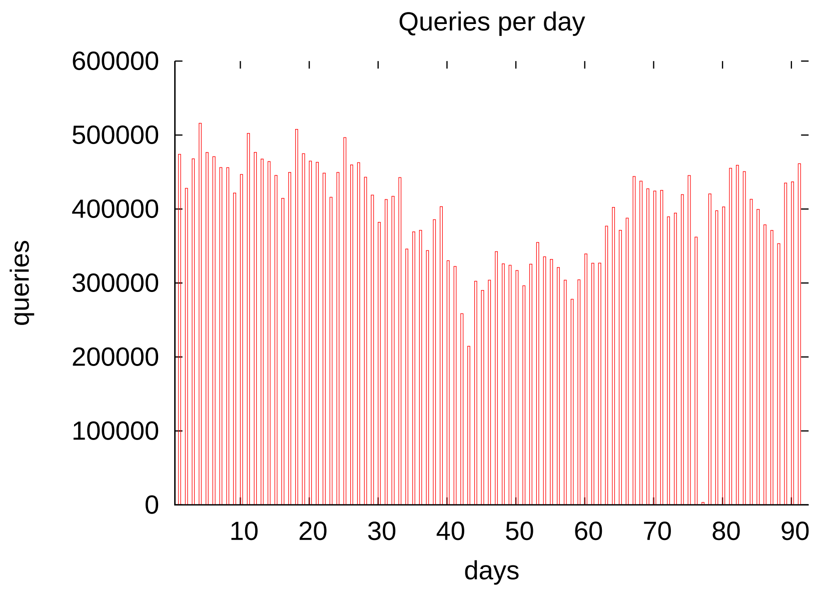


Figure 5.1: #queries per day

The first option is very unlikely, no evidence of a national holiday or disaster was found. The second could be true but still unlikely. So option three must have been the case, although no evidence was found to support it.

5.5 Conclusion

Concluding, there is no clear evidence, nor an other good reason why the AOL search data should be classified as unreliable.

Chapter 6

Linguistic Experiments

6.1 Introduction

Now that the usability of the AOL search data has been acknowledged in chapter 5, it is time to start processing the actual data. Before we can work toward answering this research's main question, there is one step that needs to be taken first. As the title of this thesis predicts, we will look at the AOL search data through and Information Retrieval view. To be a bit more specific, by using the theory behind Formal Concept Analysis we have seen in chapter 4 and which will return in the next phase of our research in chapter 7.

Before we can do that, we should assure that it is logical to look at the data in this way. This has been briefly mentioned in chapter 3; Formal Concept Analysis is about objects with attributes and attributes that belong to objects that together form a logical construct, the concept. If we were to project that view onto the AOL search data, our main assumption is that people (the objects in our research) can be characterised by their search terms (the attributes in our research). And since these search terms are written words, it is assumed that the usage of those words could follow the same pattern as natural language.

We have seen the linguistic laws Zipf's Law and Heaps' Law in chapter 3 and now it is time to see if these can be applied to the research data.

6.2 Zipf and the AOL data

6.2.1 AOL Word Counter

Zipf's and Heaps' Law dealt with occurrences and frequencies of words. So to do the same with the AOL search data, we created a little tool to do the boring work for us. The tool, dubbed AOL Word Counter, was written in Java. The program (as well as the other tools created for the research), including source code and documentation will be available at `www.navcon.info`.

AOL Word Counter uses a quite simple algorithm. It reads in the original data files, one line at a time. It then splits each line in the different columns, although it will not use all columns, since we are only interested at the query field at this time. Subsequently the query is split into separate words and those words are kept/added to a word list, along with the number of occurrences. At the end, this word list is outputted into a simple file. This output file contains nothing more than the words plus their occurrences.

6.2.2 Test Results

The words themselves in the output file have no real meaning, it is the number of times they occur that we are interested in. Therefore we strip the first column, containing the words, order the second column descending and add line numbers before them. This can all be done with simple Unix-commands. The file that is left can then be processed by a plotting program like gnuplot to produce the following diagram:

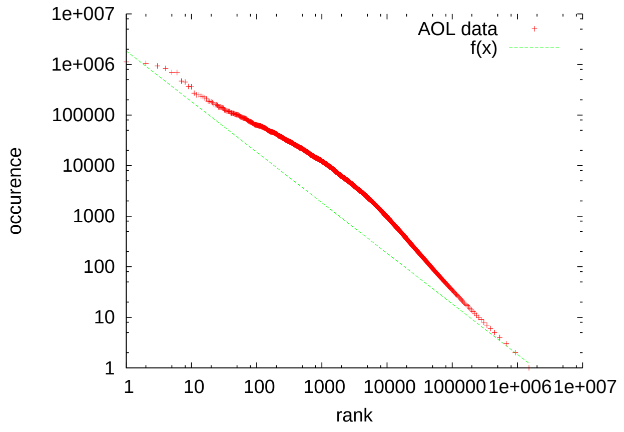


Figure 6.1: The occurrence of words in the queries

6.2.3 Mathematical

As we can see in figure 6.1 the AOL search data really has almost the same pattern as Zipf predicted for natural language, displayed in green. As a reminder, this was the formula for Zipf's Law:

$$f(k;s,N) = \frac{1/k^s}{\sum_{n=1}^{N} 1/n^s}$$

In the case of the AOL data, if we approximate the formula, the variable s is 0.480256 and the variable N is 3849555.

6.3 Heaps and the AOL data

6.3.1 Modifying AOL Word Counter

The first version of AOL Word Counter was designed specifically to acquire data to possibly support Zipf's Law. But since Heaps' and Zipf's Law are related [5] it is not hard to believe that the tool could serve another purpose too.

We adjusted AOL Word Counter to also keep a record of the number of lines read up to a certain point and the vocabulary size at that point. It does not have to do that with each line as the output file would become as large as the input files, which is larger than necessary. So it now saves that data every 1000 lines and at the end it outputs a second file, containing the number of lines processed and the associated vocabulary size.

6.3.2 Test Results

When we first plotted this data, again using gnuplot, we got the following diagram:

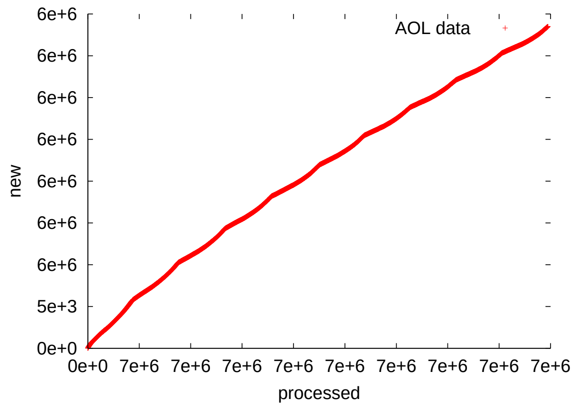


Figure 6.2: The increase of the 'vocabulary'

As you can see this is a pretty interesting line since it does not look like the line we saw in chapter 3. We see a rather stable growth although there appear to be ten little curves. The overall picture seems to be slightly sloping like the theoretical Heaps' curve. However, this is a phenomenon we cannot really explain.

At first puzzled by this result, we look at the facts: there are ten original AOL search data files; there are ten curves in our diagram. So there could be a connection. Perhaps it has something to do with the progression over time. Therefore we return to the original files, merge them together, order them on time of the query, split them again over ten files. Of course we left out the 'header' lines since they are no part of the actual data, although just ten lines probably would not have any noticeable influence. After running our tool again on these new, ordered files, we get the following diagram.

And this one looks a lot more like the one we expected. Do notice that the 'breaking point' is rather direct. It is almost as if the first part has an other word composition than the second. You would expect that there were a lot less queries after the breaking point. But if we think back to our queries per day diagram in chapter 5 we did not notice such a phenomenon. It cannot be the cause of that one 'missing day' either because that was a one time occurrence which would be visible as a one time fluctuation in the diagram after which it would resume its original course. This is not the case however. Unfortunately we cannot explain it, but this is not necessary either since it lies outside the scope of our research.

What we can possibly explain though, is the difference between the graphs in figures 6.2 and 6.3. Just as with the entire English language, a corpus, for example the Brown Corpus, is a small but representative part of it. And the assumption that it is representative is backed up with common sense and the fact that it holds (almost) the same values for the variables

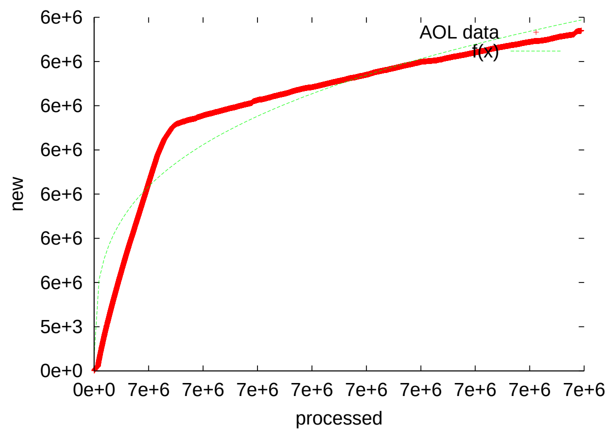


Figure 6.3: The increase of the 'vocabulary' after ordering

as other corpora. If we project this knowledge over our test results, we can assume that each of the ten original input files should be (almost) as representative for searching behaviour as the whole thing. Therefore when we plot the data in its original order, you get ten little similar curves after each other.

6.3.3 Mathematical

For completeness, we shall look at Heaps' Laws formula and see if we can determine the values for the variables in figure 6.3. As a reminder, the formula for Heaps' Law was:

$$V_R(n) = \alpha n^{\beta}$$

In the case of our research data, if we approximate the formula, the variable α is 18430.5 and the variable β is 0.293428.

6.4 Stop Words

In chapter 3 there was also a section on stop words. Excluding stop words from your data often is a good choice because they add less real information. What the experiments here show is that (the AOL) search data follows behaviour typical of natural language. Because of that we can conclude that we can dismiss several words from the data; stop words.

For this research a stop word list was used that is made by an Information Retrieval institute, especially for use with search data. This stop word list can be found on the website of the Information Retrieval Group of the University of Glasgow. All but 5 of the words on the stop word list were found in the research data and of the top 10 most used words, 8 were on the stop word list. This lead to the assumption that leaving these words out was useful. Applying the knowledge on stop words then lead to a reduction the size of the processed data. This resulted in better computability. More on computability in later chapters.

Chapter 7

Conceptual Experiments

7.1 Introduction

Chapter 4 dealt with concepts and Formal Concept Analysis (FCA). The main question for this research was whether interesting new information could be found in the form of classification of groups of people by applying FCA on the AOL search data. This chapter describes the results.

7.2 Searching for Formal Concepts

Reinventing the wheel obviously is not smart to do, so we will use the Ganter algorithm implementation by Mark Blokpoel as described in chapter 4. In order to use this program, from now on simply called Ganter, it must first be clear which data exactly is going to be used. As seen before, each line of the AOL search data consists of an id, the query, a query time, an optional clicked URL and an optional rank for this URL in the search results.

The main issue with the AOL Scandal was that the data could contain personal information about AOL's customers that should not have been released. The main assumption for this research is that search terms say something about a person. Therefore, in this experiment FCA will be applied to the id's and the query they used. As seen in chapter 4, FCA can be done on any given context, as long as there are two collections and a binary relation between them.

There is indeed a binary relation between id's and queries, because a person p has done query q or not, So the first step in this experiment is extracting the collections of persons and queries from the AOL search data. But since a query can consist of multiple words and the query 'cure headache' and 'headache cure' could be interpreted as containing the same amount of information, the queries will be split up into separate words. This is an important assumption, because it now totally ignores the relations between words and their syntactical and semantical structures. This might be an issue to address in possible future research.

7.3 AOL Reader

In order to extract the data needed for FCA from the AOL search data another tool was created, dubbed AOL Reader. Again it is written in Java and the sources are available so contact the author if you wish to have a look at them. AOL Reader reads in the original input files, strips the superfluous header lines and cuts the lines into pieces. It then stores the id and the words used in the query in a map data structure. At first it seemed easy and logical to export this mapping to a SQL structure to have better access to it so AOL Reader has a SQL module. AOL Reader later got modified to output XML-files that could serve as input for Ganter. To accomplish this goal, it uses the same XML libraries as Ganter.

In initial tests, AOL Reader outputted data of the first 100 id's it had found. This proved to be reasonably quick so it can be upscaled. Unfortunately it soon became clear that the amount of data in the AOL search data was enormous and could become a problem. For instance, AOL Reader took 8 hours and 7 minutes to process everything. But Ganter had been computing for days, producing gigabytes of data, so this calls for a plan.

7.4 Limiting to Sub Lattice

Calculability obviously is an issue. So for following experiments the searching area will be limited a bit. A sub lattice of the theoretical entire lattice may be a solution, as computability might improve and it is likely to still get worthy results because even a sub lattice of this magnitude is rather large.

With a few simple adjustments AOL Reader now takes a term as an argument and it would only process lines of data that had this term in the query. Next step is to select viable candidate terms that represent a large enough part of the data collection, but small enough to be reasonably computable. To do this simply run a simple query against the database which now also contained the mapping of id's against words, thanks to AOL Reader:

```
SELECT wordid, COUNT(*)
FROM wordsbyuid
GROUP BY wordid
HAVING COUNT(*) > 1
ORDER BY 2 DESC
INTO OUTFILE 'uidsperwordid.dat';

Query OK, 708463 rows affected (32 min 58.97 sec)
```

Several interesting limiting words came up after looking through `uidsperwordid.dat` and testing some words with different numbers of id's that have searched for them. For the remaining of the chapter, the word *nicotine* will be used. Note that this is merely an example, no possible harm was intended. The word nicotine has very representative statistical data and has an outcome that is easy to comprehend.

Statistical data about the word nicotine in the AOL data:

- There are 147 id's that used it in a query;
- These id's have a combined number of 75349 attributes;
- This results in 103031 concepts found;
- Ganter took an average 914.016 seconds to calculate this.

7.5 NavCon

Over the years, a lot of research has been done on efficient generation of concepts. Although FCA is an interesting technique and proved to be useful in many applications, the resulting conceptual structures tend to be very large for real world applications. While toy concept lattices containing 10 or 20 concepts can be visualized easily, some applications generate more than 100.000 concepts, like the nicotine sub lattice example. Obviously in these cases the need for a good navigation tool is vital.

To accommodate researchers who successfully want to use FCA on large dataset `NavCon`[1] was developed. The program's name is loosely inspired by a possible abbreviation for 'Navigation of Concepts' and spiced up to make it look like Defcon, the defense readiness condition of the United States Armed Forces.

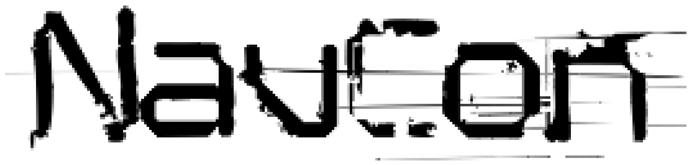


Figure 7.1: NavCon logo

The use of `NavCon` (see figure 7.2) can be split up into two phases:

1. *The calculation phase*
   In this phase (only needed to run once) `NavCon` calculates the concepts' hierarchical structure and generates `HTML` files that represent the lattice.

2. *The navigation phase*
   Using `NavCon`'s internal browser (or any other browser) the output that was created in step 1 can be used to navigate the concept lattice. A special `CSS` style file allows embedding of the output in tailor made applications.

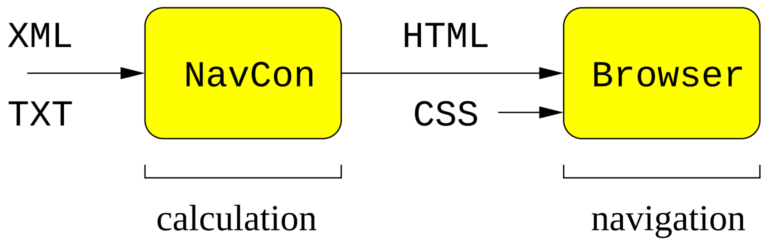


Figure 7.2: Using `NavCon`

[1] `NavCon` is developed using open source software like Java, Eclipse, SWT, Log4j. To speed up the execution of NavCon, parts of it are written in native C code using JNI.

7.5.1 Reading Concepts

NavCon was made to be able to read in Ganter output files (see figure 7.3. This is because the XML-output of Ganter is a very neat and well-constructed protocol to communicate about concepts. For larger scale contexts like ours however, there rises another size problem. The XML-output format uses quite some space for just meta-data:

- 23 characters per output file
- 21 characters per concept (excluding contents of the concept)
- 20 characters per object (excluding contents of the object)
- 26 characters per attribute (excluding contents of the attribute)

This in contrast to the TXT-output which uses

- 0 characters per output file
- 5 characters per concept (excluding contents of the concept)
- 2 characters per object (excluding contents of the object)
- 2 characters per attribute (excluding contents of the attribute)

Put this in perspective to the 103031 concepts for a very small sub lattice and it obviously is smart to use the TXT-output as output for Ganter and thus input for NavCon.

```
[mercurius, venus]:[near, small, no moon]
```

```
<concepts>
  <concept>
    <object>mercurius</object>
    <object>venus</object>
    <attribute>near</attribute>
    <attribute>small</attribute>
    <attribute>no moon</attribute>
  </concept>
</concepts>
```

Figure 7.3: Sample `TXT` and `XML` input

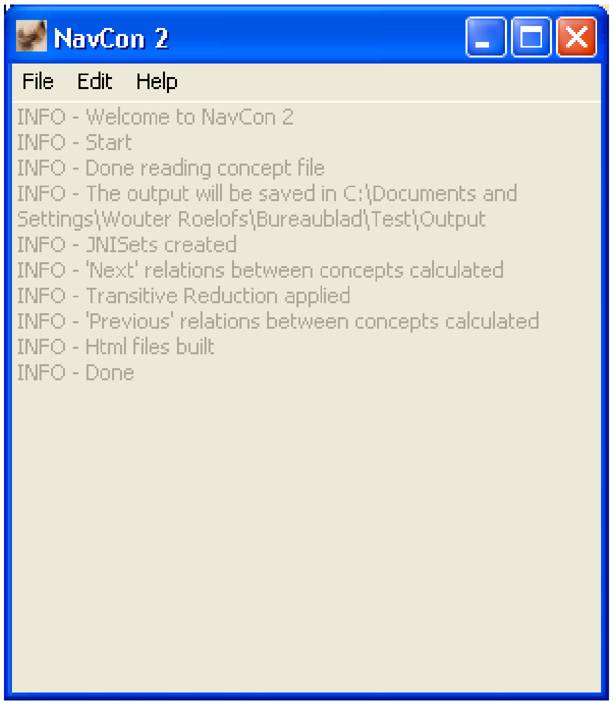
7.5.2 Reduction Engine

As mentioned in chapter 4 Blokpoel's Ganter program is quite simple and does not store any information about the relation between concepts. But in order to navigate through a lattice you need to know about the paths you can take. For that reason, most of NavCon's processing time goes to the calculation of these relations.

As seen in chapter 4 the relation that is used to define concept lattices, the $\subseteq$ relation, is defined as: Concept c_1 is smaller than concept c_2 when the attribute set of c_1 is a subset of that of c_2. It is this knowledge that has been put into NavCon. After the concepts are read in, NavCon calculates the relational structure by iterating over all concepts, adding references to all concepts larger than them. The iterating is a $O(n^3)$ complexity routine which is not too bad, but the determining whether a concept is smaller than an other concept needed to be be as efficient as possible. For this reason small pieces of fast C-code were written and used from within Java. It says that concept c_1 is a subset of concept c_2 when the attributes set of c_1 is a subset of that of c_2.

Obviously you do not need every possible subset relation in your lattice. When a concept c_1 is smaller than concept c_2 and concept c_2 is smaller than concept c_3 then concept c_1 is also smaller than concept c_3. But this last subset relation is omitted from the lattice as it is implicit and can be found by doing a transitive closure. Therefore NavCon now applies the opposite, the transitive reduction. In order to do so, it simply iterates over all concepts again, and for each 'nextConcept' it calculates whether this nextConcept is a subset of any of the other nextConcepts. If so, it removes the latter.

In a lattice you possibly want to be able to both walk up and walk down, but so far NavCon only knows about the nextConcepts for each concept. So it iterates one last time over all concepts, walking over all nextConcepts and filling 'previousConcepts' with the opposite data; whenever concept c_1 has a nextConcept c_2, it adds c_1 to the previousConcepts of c_2.



```
INFO - Welcome to NavCon 2
INFO - Start
INFO - Done reading concept file
INFO - The output will be saved in C:\Documents and
Settings\Wouter Roelofs\Bureaublad\Test\Output
INFO - JNISets created
INFO - 'Next' relations between concepts calculated
INFO - Transitive Reduction applied
INFO - 'Previous' relations between concepts calculated
INFO - Html files built
INFO - Done
```

Figure 7.4: NavCon finished calculating the lattice structure

7.5.3 Writing the Output

The calculation of the complete lattice structure is rather time and processing power consuming because of the several $O(n)$ complexity loops. It would not be smart to do this every time you want to navigate the lattice. So it is smart to think up a way to somehow store the entire lattice and make it easy to navigate. Some sort of bit stream of the data can be extracted and store it on your hard disk. But this would be an unreadable file to humans and still requires a program to read it in again and somehow display it to the user.

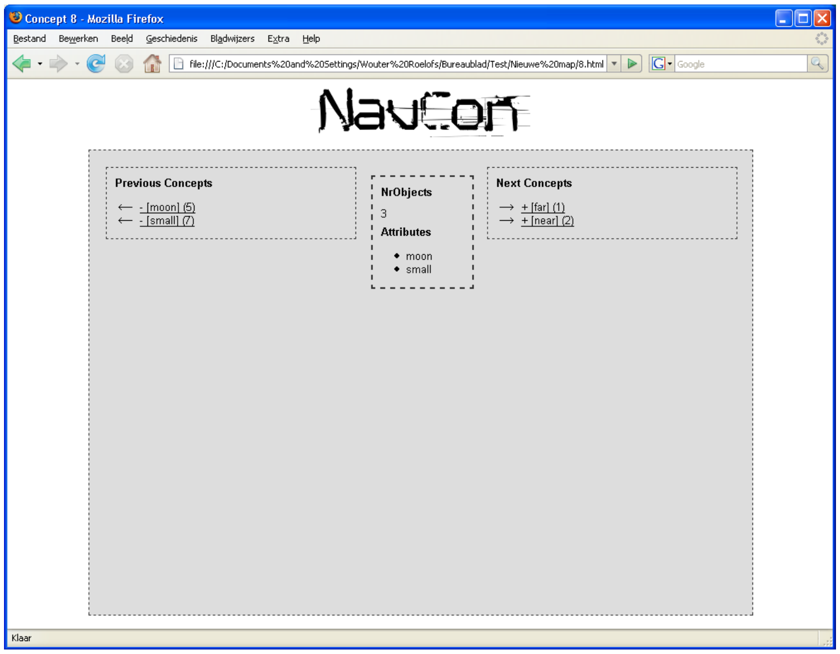
So it is imperative to store the information on the concepts somehow and make it possible to easily walk through the lattice. A technique that already exists comes to mind, namely HTML. HTML provides a way to link from one file to another using hyperlinks and the nicest part is that there is no need for special software to use it; every computer has a browser.

NavCon therefore outputs HTML files, one for every concept. The same trap as Ganter's XML output format lingers, which is large meta-data for each concept. That is why NavCon produces a very basic HTML structure that still is fully HTML compliant. When processing the nicotine sub lattice with NavCon 103032 HTML files (103031 concepts and 1 index file) are produced with an average size of 3110 bytes.

Using `HTML` as output markup language has several advantages:

- No special software needed to view the result: any ordinary browser will do.
- Fast lookups: just one `HTML` file loaded at a time with a very small footprint.
- Navigational abilities through hyperlinks, bookmarks (favorites) for interesting places, back and forward semantics using the browser's buttons.
- Styleabality and seamless integration in other applications through `CSS`.
- Scalability: linear in the number of concepts.

See figure 7.5 for an overview on what output made by NavCon looks like.


Concept 8 - Mozilla Firefox
Bestand
Bewerken
Beeld
Geschiedenis
Bladwijzers
Extra
Help
file:///C:/Documents%20and%20Settings/Wouter%20Roelofs/Bureaublad/Test/Nieuwe%20map/8.html
Google
NavCon
Klaar

Previous Concepts

← - [moon] (5)
← - [small] (7)

NrObjects

3

Attributes

- moon
- small

Next Concepts

→ + [far] (1)
→ + [near] (2)

Figure 7.5: Sample navigation

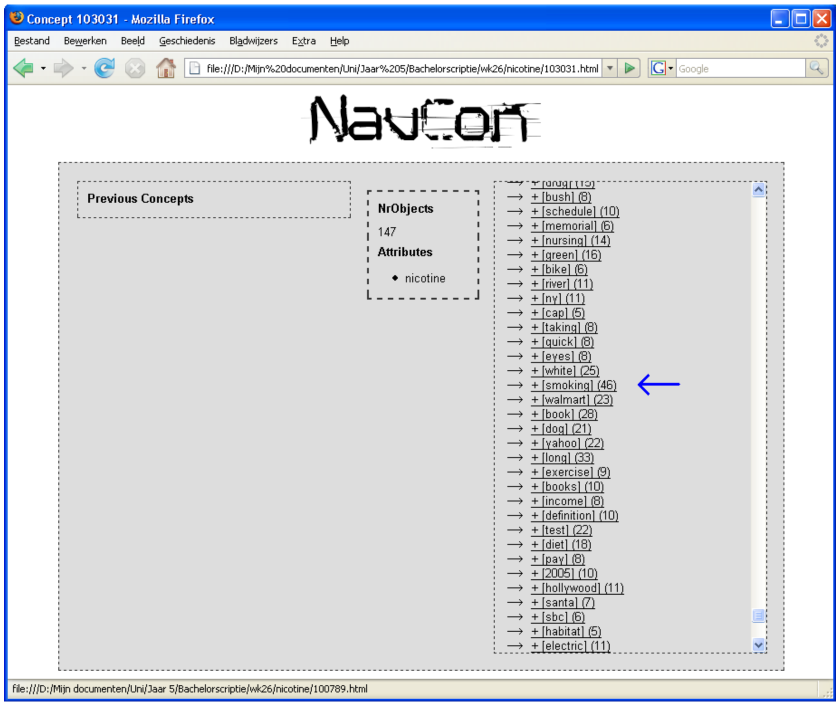
As you can see NavCon displays very specific information. The current concept is displayed in the middle in the box with the thicker border. The box on the left represents the previous concepts and the box on the right the next concepts. NavCon is highly configurable when it comes to displaying the information on the concept lattice. In this specific example the default setup has been chosen, omitting the actual objects from the HTML files, replacing it with simply the number of objects in the current concept. The attributes are shown below the number of objects. Both the upward and downward navigation contain limited information. Just the attributes that are added to the current set when navigating to a next concept are displayed, along with the number of objects on that next concept. This saves a lot of space, especially when the navigation has already been through a few steps. You can imagine that if all the attributes were displayed in the links, when there are for instance 7 or more per concept, the whole thing gets really cluttered, undermining the aspect of efficiently visualizing the lattice.

7.6 Conclusion

To conclude this chapter and the research, a demonstration navigation will be presented on the AOL search data to show that the formal concepts formed can be interpreted as useful data, thus answering the research's third sub question positively. See figures 7.6 to 7.9 for the demonstration.

For the demonstration, again, the standard styling is used. This style seems to work really well for handling large data sets, like this AOL search data. Figure 7.6 shows the initial state, starting at the bottom concept containing just the attribute *nicotine* and having 147 objects, which is the total number of objects in this sub lattice. When for instance clicking on the concept that adds *smoking* (closely related to nicotine of course), denoted by the blue arrow, you arrive at figure 7.7. Step 2 has 46 objects as seen in step 1 when clicking on the next concept. Notice that there is a way back down now, through the removal of *smoking*. Also note that the list of next concepts is fairly long. This is explainable as a lattice usually is diamond shaped. Toward the middle there are more and more concepts, after that the number decreases. After clicking on the concept that has the word *disease* extra in its attribute set, you arrive at step 3 (figure 7.8. You immediately notice that the number of next concepts is much smaller now as you reach the top of the lattice. Now something interesting is revealed: among the next concepts there is one that has more than one attribute more than the current one. Somehow, through the searchers' use of queries, certain keyword clutter together. This is exactly what is expected out of FCA and hoped for this research! Finally, figure 7.9 shows the concept that has the attribute set {center, nicotine, medical, smoking, disease, nursing, healthcare}. Steps toward next concepts seem to have little additional value. As you reach the top, the number of objects is decreased to a minimum, leaving bags of keywords that cannot really be interpreted anymore as they almost specifically target individuals.

This concludes these experiments and indicate that it is indeed possible to get valid results after applying FCA to a live search data set.


Concept 103031 - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
file:///D:/Mijn%20documenten/Uni/Jaar%205/Bachelorscriptie/wk26/nicotine/103031.html
Google
NavCon

Previous Concepts

NrObjects

147

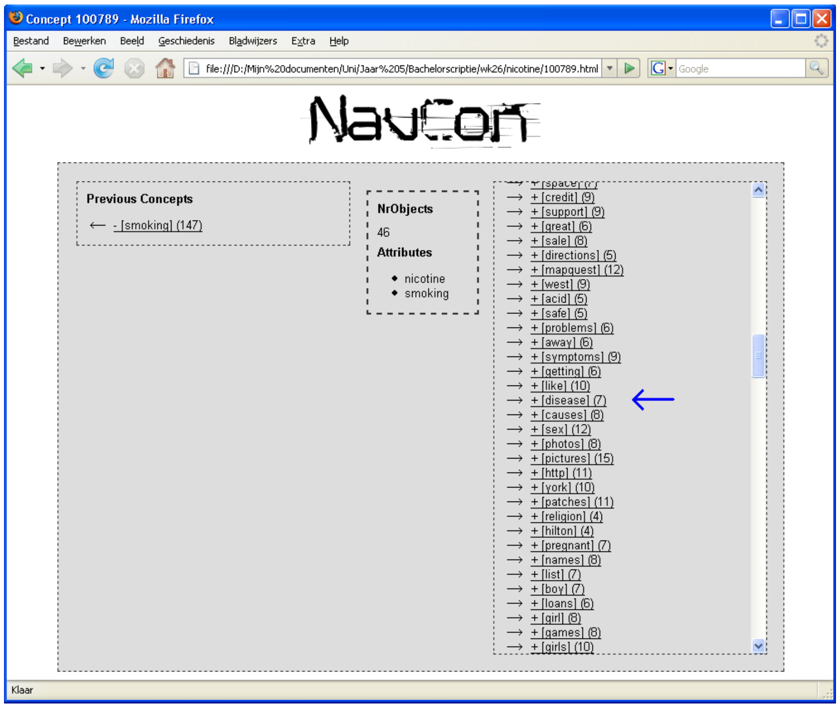
Attributes

- nicotine

→ + [bush] (8)
→ + [schedule] (10)
→ + [memorial] (6)
→ + [nursing] (14)
→ + [green] (16)
→ + [bike] (6)
→ + [river] (11)
→ + [ny] (11)
→ + [cap] (5)
→ + [taking] (8)
→ + [quick] (8)
→ + [eyes] (8)
→ + [white] (25)
→ + [smoking] (46) ←
→ + [walmart] (23)
→ + [book] (28)
→ + [dog] (21)
→ + [yahoo] (22)
→ + [long] (33)
→ + [exercise] (9)
→ + [books] (10)
→ + [income] (8)
→ + [definition] (10)
→ + [test] (22)
→ + [diet] (18)
→ + [pay] (8)
→ + [2005] (10)
→ + [hollywood] (11)
→ + [santa] (7)
→ + [sbc] (6)
→ + [habitat] (5)
→ + [electric] (11)

file:///D:/Mijn documenten/Uni/Jaar 5/Bachelorscriptie/wk26/nicotine/100789.html

Figure 7.6: Step 1 of demo navigation through the AOL data


Concept 100789 - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
file:///D:/Mijn%20documenten/Uni/Jaar%205/Bachelorscriptie/wk26/nicotine/100789.html
Google
NavCon

Previous Concepts

← - [smoking] (147)

NrObjects

46

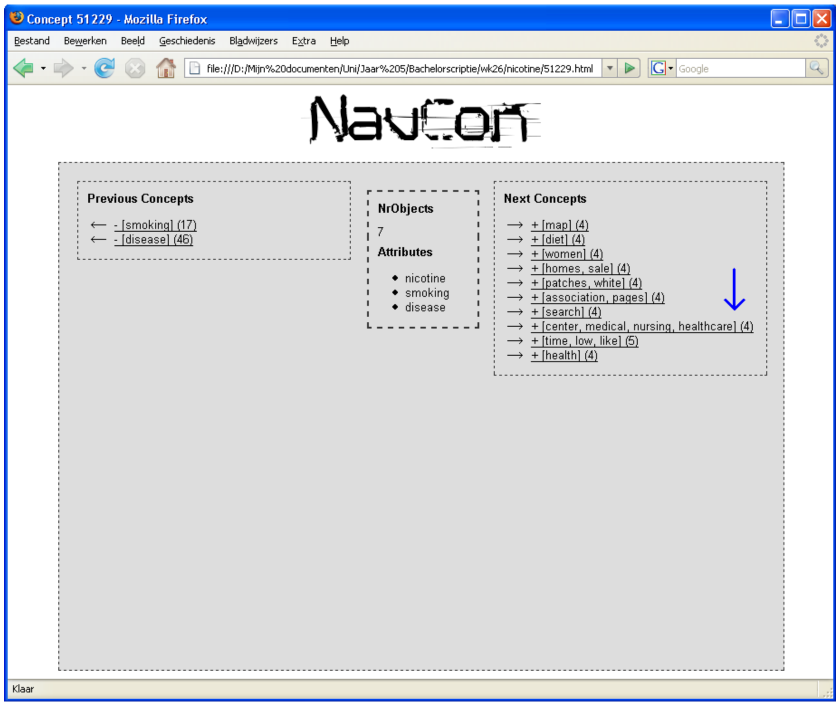
Attributes

- nicotine
- smoking

→ + [credit] (9)
→ + [support] (9)
→ + [great] (6)
→ + [sale] (8)
→ + [directions] (5)
→ + [mapquest] (12)
→ + [west] (9)
→ + [acid] (5)
→ + [safe] (5)
→ + [problems] (6)
→ + [away] (6)
→ + [symptoms] (9)
→ + [getting] (6)
→ + [like] (10)
→ + [disease] (7) ←
→ + [causes] (8)
→ + [sex] (12)
→ + [photos] (8)
→ + [pictures] (15)
→ + [http] (11)
→ + [york] (10)
→ + [patches] (11)
→ + [religion] (4)
→ + [hilton] (4)
→ + [pregnant] (7)
→ + [names] (8)
→ + [list] (7)
→ + [boy] (7)
→ + [loans] (6)
→ + [girl] (8)
→ + [games] (8)
→ + [girls] (10)

Klaar

Figure 7.7: Step 2 of demo navigation through the AOL data


Concept 51229 - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
Google
NavCon
Klaar

file:///D:/Mijn%20documenten/Uni/Jaar%205/Bachelorscriptie/wk26/nicotine/51229.html

Previous Concepts

← - [smoking] (17)
← - [disease] (46)

NrObjects

7

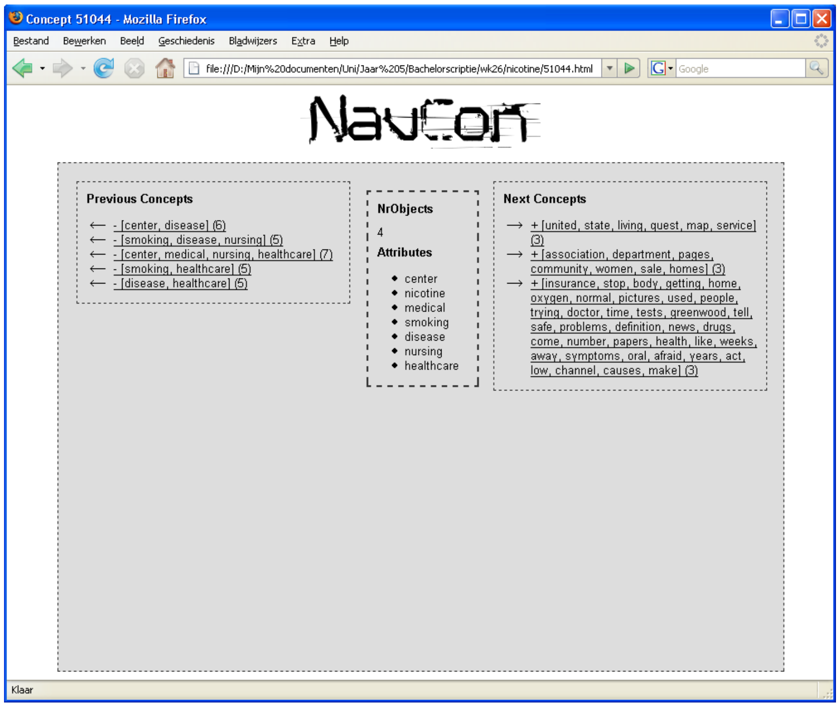
Attributes

- nicotine
- smoking
- disease

Next Concepts

→ + [map] (4)
→ + [diet] (4)
→ + [women] (4)
→ + [homes, sale] (4)
→ + [patches, white] (4)
→ + [association, pages] (4)
→ + [search] (4)
→ + [center, medical, nursing, healthcare] (4)
→ + [time, low, like] (5)
→ + [health] (4)

Figure 7.8: Step 3 of demo navigation through the AOL data


Concept 51044 - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
Google
NavCon
Klaar

file:///D:/Mijn%20documenten/Uni/Jaar%205/Bachelorscriptie/wk26/nicotine/51044.html

Previous Concepts

← - [center, disease] (6)
← - [smoking, disease, nursing] (5)
← - [center, medical, nursing, healthcare] (7)
← - [smoking, healthcare] (5)
← - [disease, healthcare] (5)

NrObjects

4

Attributes

- center
- nicotine
- medical
- smoking
- disease
- nursing
- healthcare

Next Concepts

→ + [united, state, living, quest, map, service] (3)
→ + [association, department, pages, community, women, sale, homes] (3)
→ + [insurance, stop, body, getting, home, oxygen, normal, pictures, used, people, trying, doctor, time, tests, greenwood, tell, safe, problems, definition, news, drugs, come, number, papers, health, like, weeks, away, symptoms, oral, afraid, years, act, low, channel, causes, make] (3)

Figure 7.9: Step 4 of demo navigation through the AOL data

Chapter 8

Conclusions

Most modern Information Retrieval systems are still keyword-based. Although this may seem a very simplistic way of looking at documents and information need, the results of these systems are good. Most of their strength comes out of the fact that each keyword represents a concept in human minds [6]. It is this very notion that forms the foundation of this research. If people use keywords to translate their thoughts into search queries, these same keywords are bound to say something about the people.

In this research, an attempt has been done at showing that this assumption is a plausible one. This has been done using some of the knowledge on Search Engines, Computational Linguistics and Formal Concept Analysis (FCA). Applying this to the hot issue of the so-called AOL Scandal the following conclusions can be made.

8.1 Conclusion 1

Is the integrity of the data good enough to support valid outcomes?

As shown in chapter 5 the AOL search data

- seems to be consistent with AOL's claims
- has a logically explainable weekly rhythm
- has an unexplained anomaly that does not seem to affect the integrity

Therefore it can be concluded that the data seems to have a high enough integrity to support research.

8.2 Conclusion 2

Is there an indication that the data supports FCA?

As shown in chapters 3 and 6 search keywords

- seem to obey some laws that natural language obeys
- support some assumptions to FCA as shown in [6].

The fact that Zipf's and Heaps' Law were both found in the search data is pretty amazing. Because natural language is not used for searching but simple keywords, it is expected to have characteristics of its own. But this discovery does in a way support the main assumption that keywords used by a searcher say something about that searcher.

8.3 Conclusion 3

Does applying FCA to the data lead to relevant results?

As shown in chapters 4 and 7 the AOL search data

- is too large to handle at once, but can effectively be split up into sub lattices
- as a sub lattice is still too large to visualize and navigate through using existing techniques
- can be visualized and navigated using NavCon
- leads to some interpretable concepts

Applying FCA to the research data indeed seems to lead to relevant results. Initially a lot more was expected, but the sheer size of the data lead to the necessity of creating NavCon. This however made a contribution to the Information Retrieval world in the form of a tool capable of handling enormous collections of concepts.

8.4 Main Conclusion

Is it possible to classify groups of people by their searching behaviour in an automated way using Formal Concept Analysis?

The results of the experiments may not be as shocking as they could have, but still a lot of work has been done. Running into trouble with the computability of the data, NavCon was made. With this tool it has been shown that applying FCA to search data can indeed be used to find natural groups of people. Although this result may seem insignificant, it is a foundation to possible future research on this subject. The next section will shortly address possible future steps.

8.5 Future Research

As with all good academic research, the scope of this thesis has been revised numerous times. The final research question looks quite different from the initial one and many concessions have been made. This chapter is an extra chapter to address some of the original thoughts on the research that were left out due to size and time constraints. Two things will be discussed:

1. Possible enhancements to the algorithm
2. Possible practical purposes this research can serve

8.5.1 Applying Heuristics

The main problem when trying to interpret formal concepts is that people try to relate it to things they know. Even when looking at very abstract formal concepts like the example used in chapter 4, ({mars, earth},{moon, near, small}), humans like to label it. If only it has a name, it is tangible, therefore easier to comprehend.

FCA applied to the AOL search data is not. Practically no concept can be labelled, at least not in a unambiguous way. But sometimes it can be very useful to do so, for example in the case of this research and probably also in future research on search data. Therefore future research could focus on some heuristics to help in the process. A machine can probably never label concepts on its own, but it could assist the human user.

Weighing the Concepts

One possible answer to this problem is adding weights to concepts. If a concept has some sort of weight, a concept c_1 can be called less or more important than a concept c_2. This can help identifying 'important' concepts. Using the linked structure of a lattice, weights can be calculated for every concept given just little data. And cumulative weights can be calculated too, both upward and downward.

There is an ongoing research at the time of writing that could have made use of this extension. It deals with author identification through chat logs using FCA [13].

Weighted concepts could also prove to be useful when categorizing people through search data. If there are values for certain keywords, these can be used to calculate total scores for concepts. The number of objects, the number of attributes and the position in lattice could also be taken into consideration. Then this total value could indicate some concepts to be more important than others. This can obviously be of great help when browsing through hundreds of thousands of concepts.

Other Fields of Research

Other fields of research can possibly contribute to future research too. This is quite obvious but nevertheless stated for completeness. Perhaps Psychology can add useful knowledge on how people think (and search). This research's main assumption of keywords saying something about people could be backed up by Psychology too. Then there is Linguistic Sciences that look at Information Retrieval from an other angle than Information Sciences. Both may be able to contribute to future research.

8.5.2 Practical Solutions

This research was partially theoretical and partially experimental. The possibilities for useful results have been examined, not taken into consideration whether this knowledge can be put in practise. This is exactly what this addendum is about.

Possible fields of interest when thinking about practical appliance of this research:

- Advertisement The advertising world obviously has a large interest in categorizing individuals. There already are efforts on using keywords to personalize advertisements on the internet and this research could improve this.

- Health Care Imagine someone having a strange rash on his skin that irritates a little but not enough for him/her to go to the doctor. What if this is a symptom of a virus that can spread through the air and affects a lot of people. That person repeatedly searches on the internet for clues on what his/her rash may be. It could be possible in the future that this person get identified and contacted by the hospital in an early stage. The same could go for depressive people that have a risk to hurt themselves.
- Fighting Crime A similar example to the health care example: imagine a criminal planning an attack. Somehow his plans can be extracted from this search behaviour. It is not hard to see the advantages in stopping this attack from happening.

And there are many more examples which are probably not all as welcome as others.

8.5.3 Availability & Computability

A small last side node is that of availability and computability. Through the 'scandal' of AOL a small portion of search data has become accessible, but this already is outdated for the purposes just described. Real large amounts of search data is in the hands of Google or other mayor search engines.

And if the data is available, then there is the problem of calculating with it. This research has shown that even a reasonably sized collection already needs vast amounts of processing capability. This issue could perhaps be addressed by using a form of incremental FCA.

Appendix A

The AOL Search Data in SQL

In the first part of our research, the AOL data was put into a SQL database. This appendix describes the structure of that database and the method that was used to import the data into the database.

Note that the queries shown in this appendix were executed on a AMD Athlon XP2600+ system with 1GB of ram. The time queries took may therefore differ on your system.

In chapter 5 a part of the file `U500k_README.txt` was mentioned containing some 'facts' on the AOL search data. In an other part of the file AOL gave an overview of the structure of the data:

```
The data set includes {AnonID, Query, QueryTime, ItemRank,
ClickURL}.
AnonID - an anonymous user ID number.
Query  - the query issued by the user, case shifted with
         most punctuation removed.
QueryTime - the time at which the query was submitted for search.
ItemRank  - if the user clicked on a search result, the rank of
            the item on which they clicked is listed.
ClickURL  - if the user clicked on a search result, the domain
            portion of the URL in the clicked result is listed.
```

Some of this data was already implicitly used during this thesis. In order to copy the data from the text files into a SQL database we first create the table, assuming there already is a working SQL server, database and an account with enough rights:

```
mysql> CREATE TABLE aoldata (
    ->  anonid INT UNSIGNED NOT null,
    ->  query VARCHAR(255),
    ->  querytime DATETIME,
    ->  itemrank INT UNSIGNED,
    ->  clickurl VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

Now we load the text files into this database by executing the following query for each input file:

```
mysql> LOAD DATA LOCAL INFILE 'user-ct-test-collection-01.txt'
    -> INTO TABLE aoldata
    -> FIELDS TERMINATED BY '\t'
    -> LINES TERMINATED BY '\n'
    -> (anonid, query, querytime, itemrank, clickurl);
Query OK, 3558412 rows affected (18.44 sec)
```

In chapter 5 we saw that the original input files contain header lines with faked data. These lines were obviously copied to the SQL database so must therefore be deleted.

```
mysql> SELECT *
    -> FROM data
    -> WHERE anonid=0;
+--------+-----------+---------------------+----------+----------------------+
| anonid | query     | querytime           | itemrank | clickurl             |
+--------+-----------+---------------------+----------+----------------------+
|      0 | Query     | 0000-00-00 00:00:00 |        0 | ClickURL             |
|      0 | Query     | 0000-00-00 00:00:00 |        0 | ClickURL             |
|      0 | Query     | 0000-00-00 00:00:00 |        0 | ClickURL             |
|      0 | Query     | 0000-00-00 00:00:00 |        0 | ClickURL             |
|      0 | Query     | 0000-00-00 00:00:00 |        0 | ClickURL             |
|      0 | Query     | 0000-00-00 00:00:00 |        0 | ClickURL             |
|      0 | Query     | 0000-00-00 00:00:00 |        0 | ClickURL             |
|      0 | Query     | 0000-00-00 00:00:00 |        0 | ClickURL             |
|      0 | match.com | 2006-03-31 06:55:53 |        2 | http://www.match.com |
|      0 | Query     | 0000-00-00 00:00:00 |        0 | ClickURL             |
|      0 | Query     | 0000-00-00 00:00:00 |        0 | ClickURL             |
+--------+-----------+---------------------+----------+----------------------+

11 rows in set (1 min 58.03 sec)
```

Notice that there is one line that seems to be a serious line of data, although it has an anonid of 0. So we should watch out what to delete.

```
mysql> DELETE FROM data
    -> WHERE clickurl=ClickURL;
Query OK, 10 rows affected (2 min 2.08 sec)
```

This was the final step, the SQL database is now ready for use. Note the simplicity of use; when you would like to know which exact same query was executed most, you can now use the power of SQL:

```
SELECT query, COUNT(*)
FROM data
GROUP BY query
ORDER BY 2;
```

Bibliography

[1] Mark Blokpoel. Yanacona: Ganter and parallelism, June 2006.

[2] C. Torgeson G. Pass, A. Chowdhury. A picture of search. In *The First International Conference on Scalable Information Systems*, Hong Kong, June 2006.

[3] B. Ganter. Two basic algorithms in concept analysis. Technical Report FB4-Preprint No. 831, TH Darmstadt, 1984.

[4] F.A. Grootjen and Th. P. van der Weide. Conceptual relevance feedback. In *Proceedings of the 2002 IEEE International Conference on Systems, Man and Cybernetics, (NLPKE 2002)*, Tunis, October 2002.

[5] F.A. Grootjen, D.C. van Leijenhorst, and Th. P. van der Weide. A formal derivation of Heaps' law. Technical Report NIII-R0302, University of Nijmegen, 2003.

[6] Franc Grootjen. *A Pragmatic Approach to the Conceptualisation of Language*. PhD thesis, Radboud University, Nijmegen, the Netherlands, 2005.

[7] J. Heaps. *Information Retrieval - Computational and Theoretical Aspects*. Academic Press, Inc., New York, 1978.

[8] A. Kornai. How many words are there? *Glottometrics*, 4:61–68, 2002.

[9] W. Li. Random texts exhibit Zipf's-law-like word frequency distribution. *IEEE Transactions on Information Theory*, 38(6):1842–1845, 1992.

[10] B. Mandelbrot. The pareto-levy law and the distribution of income. *International Economic Review, I*, pages 79–106, 1960.

[11] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[12] J.R.R. Tolkien. *The Lord of the Rings*. Houghton Mifflin Co., Boston, 1965.

[13] Luuk van der Knaap. Author identification in chatlogs using formal concept analysis, June 2007.

[14] R. Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470. D. Reidel Publishing Company, Dordrecht–Boston, 1982.

[15] G. Xavier. Zipf's law for cities: An explanation. *Journal of Economics*, 113(3 (August)):739–767., 1999.

[16] G. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA, 1949.