University of Groningen
Faculty of Artificial Intelligence

Master of Science Thesis

Data mining on historical access data for an intelligent caching and replacement algorithm in a digital archive of seismic data

by

Jacob Willem Gerritsen, s1211773

Supervisors RuG: prof. dr. M. Aiello and dr. L.C. Verbrugge

Supervisors Shell: J.P.M. Versmissen and J.E. van Roekel

The Hague, June 2007

Summary

This thesis focuses on the domain of digital archiving of seismic data within Shell. One of the main activities of Shell is to explore and produce oil and natural gas to meet the world's energy demand.

The digital archive consists of two storage layers. Data enters and leaves the archive via the primary layer. Moving data between the storage layers, called data migration, is organised by means of a replacement algorithm. A replacement algorithm decides upon the content of the different storage layers. In this case, the Least Recently Used (LRU) algorithm is used. This algorithm keeps the most recently used files on primary storage. The LRU algorithm is a 're-active' algorithm in the sense that it only migrates data to higher storage when a user accesses the data.

The situation as within Shell is characterised by 'active archiving'. A user actively logs on the for (de-)archiving dedicated server. Active archiving implies that just archived files will not be accessed soon; files are purposefully transferred from active to passive storage. Furthermore, de-archived files are transferred from passive to active storage. Hence, these files will probably not be de-archived soon again.

It is argued that for such a situation, the LRU algorithm is not the most appropriate replacement algorithm. It is expected that the performance can improve by replacing the algorithm.

It is proposed to formulate a 'pro-active' prefetching algorithm. The algorithm considers relative attractiveness of files, caused by relations between files. This 'knowledge' is extracted by means of data mining techniques. Clustering is used to measure general (relative) attractiveness. Association rule mining is used for analysing frequently observed joint accesses. As addition to former researches, this project takes into account generalised knowledge for prefetching.

The results show that relevant and statistically significant relations between data sets exist. These relations are extracted on different levels of data taxonomy for generalisation purposes.

The knowledge is used in prefetching algorithm DMP. The DMP algorithm

does not outperform the LRU algorithm, neither in both in terms of hit rate nor in terms of algorithm overhead. This is caused by a shift in usage with he transition from the training to the testing period. Another cause is the static nature of the used data mining techniques. Furthermore, further generalisation of association rules was impossible, therewith decreasing the value of the rules.

It is concluded that the use of results of data mining in this form cannot benefit the performance of the digital archive compared to the LRU algorithm. Therefore, the DMP algorithm must not replace the LRU algorithm at the Shell digital archive of seismic data.

Acknowledgements

First of all, my sincere thanks go to Shell, the company that has funded and supervised this research. Although the original question was completely different, they gave me the opportunity to dive deeper into the material and to finalise my second study. From the team I worked for, I would explicitly like to thank George Versmissen and Jaco van Roekel as official research supervisors. Additionally, my thanks go to Fokko Masselink and Paul Broughton from the same team within Shell. They helped me with the technical details of respectively the (spatial) database techniques and the extraction of historical access data from the digital archive.

Secondly, my sincere thanks go to the supervisors from the University of Groningen: prof. dr. Marco Aiello and dr. Rineke Verbrugge. They showed their interest in my research topic and guided me through the process of the design, execution and wrapping up of this research project. I had some trouble starting up and defining a proper research proposal. They showed their confidence, which made me go on to finish this project appropriately.

Contents

Chapter 1

Introduction

This thesis challenges the use of the Least Recently Used (LRU) algorithm as replacement algorithm for a large, digital archive of seismic data. A replacement algorithm determines what the content is of different storage layers in a layered storage environment as is a digital archive.

The LRU algorithm is the algorithm that is currently used for the digital archive at Shell[1]. The LRU only considers the last accesses of files to decide upon which files should be present on primary storage. It keeps the most recently used files on disk until the maximum capacity of primary storage is reached. Other files are replaced and migrated to lower storage. Furthermore, the LRU algorithm is a 're-active' algorithm: it only migrates data from secondary to primary storage when files are accessed by a user.

Former research shows the potential of pro-actively prefetching files to improve the hit rate of a layered storage system. These researches mainly focuss on the domain of web-page (cache) replacement algorithms.

Prefetching is based on the (relative) likelihood of files being accessed soon. Research shows the potential of using a general (relative) likelihood. Furthermore, it shows the potential of prefetching based on frequently observed joint accesses. This 'knowledge' is extracted by means of data mining.

This research uses the concept of prefetching files based on the results of data mining. However, it does so in an area in which it has not been implemented yet: digital archiving.

A feature of archiving is that few files are accessed (both in absolute terms and in ratio to the total size of the archive). That increases the difficulty of extracting useful knowledge. Therefore, an additional element of data mining is used that has not been implemented in a prefetching algorithm yet: generali-

[1] Shell is a privately owned company that strives to fulfil energy demand worldwide. Among the activities, one can find the exploration and production of hydrocarbons, i.e. oil and natural gas fields. For further information, see http://www.shell.com.

sation of data mining results. The basis for generalisation is data taxonomy. In this case, a special generalisation effort is made on spatial distance.

What remains left after having conducted this research is to incorporate dynamism of knowledge. Knowledge can alter over time. Furthermore, knowledge can differ when one to extracts knowledge during different period lengths.

Another remaining research area is to incorporate the new prefetching methods *next to* the LRU algorithm instead as a replacement of. Prefetching was already combined with the LRU algorithm in former research. We have not combined the new prefetching method based on *generalised* knowledge with the LRU algorithm.

1.1 Thesis Overview

This thesis is organised as follows. Chapter 2 describes the current organization of digital archiving of seismic data within Shell. It considers the business processes surrounding the archive and the way data migration is organised currently.

Chapter 3 challenges the Least Recently Used algorithm, which is currently used as replacement algorithm for the digital archive. Furthermore, it motivates why data mining could improve the performance when the results are used in a prefetching algorithm.

Chapter 4 describes the foundations data mining. Furthermore, the Chapter decides upon which data mining techniques are most appropriate to extract knowledge from historical access data regarding the digital archive of seismic data.

Chapter 5 describes the experiments that are conducted with the data mining techniques clustering and association rule mining.

Chapter 6 providing the knowledge that is extracted by the data mining techniques. Furthermore, it uses these results to formulate a prefetching algorithm called DMP. Subsequently, we compare the performance of this algorithm with the Least Recently Used algorithm by means of a simulation.

Chapter 7 concludes the thesis through answering whether prefetching based on the results of data mining can improve the performance of the digital archive. Furthermore, a discussion is included to review the research and choices that are made during the project.

Chapter 2

The Digital Seismic Data Archive of Shell

2.1 Introduction

The goal of this chapter is to introduce the situation within Shell regarding digital archiving of seismic data. This chapter serves as descriptive chapter and is used to challenge the LRU replacement algorithm in the next chapter.

Section 2.2 introduces the world of Shell, seismic data and digital archiving thereof. The archive within Shell uses the Least Recently Used replacement algorithm, introduced and described in Section 2.3.

2.2 Shell Seismic Digital Archive

Shell Exploration and Production is one of the major Business Units (BU) of the Shell Group[1]. One of the main goals of this BU is the exploration of hydrocarbons[2]. Shell is heavily concerned in analysing seismic data for that purpose. Seismic data represents the subsurface of the earth, both land and marine based.

2.2.1 Seismic Data Acquisition

Seismic data is collected by sending acoustic signals into the ground. This can be done by the explosion of dynamite (land acquisition) or by the production of high pressured waves (marine acquisition). The transition of rock layers in the earth causes part of the signal to continue and part of the signal to reflect. Reflections reach sensors at the surface.

[1] The Shell Group refers to the collection of publicly owned legal entities that together operate under the flag of Shell.

[2] Hydorcarbons is the collective term to describe oil and gas, which have a comparable creation process.

2.2.2 Seismic Imaging and Processing

By the presence or absence of (transitions of) rock layers, one is able to estimate the location of hydrocarbons. Making images out of seismic data is called seismic imaging; running the algorithms to do so is called seismic processing.

Seismic processing consists of applying several algorithms, which all have their benefits and shortcomings. Some are required to be run on all data sets to make an image, some are optional and some even rely on the results of other algorithms. One of the required processing activities is called 'identing'. Identing is a process of adding a coordinate reference system to raw data. It is a very precise, time consuming process.

2.2.3 Seismic Data Archiving

Seismic data is of high value to a company like Shell. It forms the first input for finding new reservoirs of oil or natural gas. There are two major reasons to archive seismic data. Archiving as a business process is 'preserving data against all imaginable contingencies'.

Firstly, certain seismic data is not acquirable anymore. The acquisition of seismic data requires the surroundings to be abandoned as much as possible. Hence, certain spots are impossible to acquire new data from due to construction of urban environments. Furthermore, issues arise when data acquisition is required in environmentally sensitive areas. For example, licenses will not be granted anymore for data acquisition in the Waddenzee.

Secondly, the nature of seismic processing causes certain steps of processing to be superfluous. The (intermediate) results can be re-used for further seismic processing efforts. Therefore, Shell maintains the digital archive for both raw data, processing results and several intermediate steps. Using (intermediate) results instead of raw data can reduce project execution time heavily, sometimes up to months.

2.2.4 Shell Digital Archive

Seismic data sets are very large, up to several TeraBytes[3]. All together, the current archive contains around 500 TeraBytes of seismic data.

One of the main requirements of an archive is that data must be protected against loss for a large number of years, while not being used frequently. In this research data from a period of slightly more than a year is analysed. Within that year, only around 10% of the data is actually used (once or more often).

A common way to establish a digital archive is to use multiple layers of storage that differ in performance and costs [15]. Higher storage layers have higher speeds but are more expensive, whereas lower layers of storage are slower

[3] 1 TeraByte is equivalent to 1024 GigaBytes.

and cheaper. Data that is not used for a long period is stored on lower layers, whereas active data is stored on higher layers.

This concept is called Hierarchical Storage Management (HSM). The total cost of ownership by using HSM can be decreased to only 2% compared to only using disk storage. Furthermore, as acquisition data is commonly delivered on magnetic tape, one is required to maintain tape infrastructure. A final benefit of tape compared to disk is reliability. Whereas (enterprise) disks can be used for 4 to 5 years, data on magnetic tape can be read for around 30 years.

The HSM concept is also used by Shell to digitally archive seismic data. The archive consists of a disk environment of approximately 1 TeraByte and a magnetic tape (robotics) environment, which can currently capture up to 2 PetaBytes[4]. The total capacity depends on the used tape medium, which currently is a medium that can store 200 GB per tape cartridge. Two tape silos are used in which around 10,000 tape cartridges can be stored. The robotics environment takes care of automatically putting tape cartridges in tape drives. Six tape drives are available, all containing a native reading/writing speed of 30 MegaBytes per second.

The system acts as a very large (Unix) File System environment. Users 'just' store their files on the file system by means of copy commands from other storage environments. Software takes care of writing data from disk to tape. It releases the files from disk, while they remain visible (called stubbing) as part of the files system. When a user tries to access a file again, while it is not on disk, the software will get the file back from tape automatically. The system works completely independent, so without any human intervention.

The time it takes to find and read data from tape constrains users in their work. For example, if a data set's size is 1 TeraByte, the reading time would take already more than 1.5 hours if all six tape drives would be used. However, the configuration is such that not all drives can be used in parallel for either reading or writing one data set to not fully occupy the system for one data set. If one tape drive would be used for this project, reading it would already take up to around 10 hours. Sizes of data sets differ, but 1 TeraByte is not very large, nor is it small.

2.2.5 Active Archiving

The digital archive that Shell maintains and controls is kept separated from other computing environments and applications within Shell. The server for the digital archive is fully dedicated to the digital archive. A user explicitly chooses to archive a file and, therefore, logs on to the server only for (de-)archiving. After having finished a seismic processing project, the user transfers files from

[4] 1 PetaByte is equivalent to 1024 TeraBytes.

his active working storage to the (shared) archive (passive storage) with the goal not to use the files for a long time. In this process, no constraints regarding links between the systems exist. The files might be needed in the future during a new project, but for now the files are not required on active storage anymore.

When files are accessed to be de-archived, they are transferred from passive storage to active storage. Hence, the files are copied from the archive file system to another file system. A user will work with the files, while they are on his active storage environment.

We call this a situation of 'active archiving'. It is expected that recently archived files will not be accessed soon. Furthermore, as de-archived files are copied to working storage, it is expected that the same data is not accessed soon again. These assumptions also build on the fact that the team that uses the system is small (around 10 people) and communicates well about which files are required per project.

The situation of 'active archiving' differs from a situation in which the concept of HSM is connected to working storage. In such a situation, files are stored on lower storage layers automatically and without active confirmation by the user. In that case, uncertainty exists whether the user would still like to actively use the files soon or will not access them anymore (for a long period or at all).

2.3 Replacement Policy

The organization of data migration between different storage layers is executed by means of replacement policies[5]. Disk usage can exceed the threshold when files are added to the disk[6]. This occurs when users archive or de-archive data, as the only access point to the archive is the disk. The replacement policy must decide upon which files must be replaced when such a situation occurs. The authors in [23] define an optimal replacement policy as in Definition 2.1.

Definition 2.1 An optimal replacement policy assures that all files that are accessed at time t are present on primary storage at time t.

An optimal replacement policy requires either to predict all future accesses or keep all files on disk. The former option is impossible, whereas the latter is not desirable due to cost and reliability issues (see Section 2.2.4). Hence, an optimal replacement policy is only theoretically possible.

[5]Data migration is the movement of data between different storage layers.

[6]A disk usage threshold makes sure that files can still be added without exceeding total capacity and disk performance does not decrease significantly. At Shell, 80% of disk storage is used as a standard number.

2.3.1 The Least Recently Used Algorithm

One of the most popular replacement policies is the Least Recently Used algorithm (LRU) [15]. The LRU algorithm is also implemented at the Shell digital archive for seismic data. The LRU algorithm decides upon which files should be on primary storage by means of the time that files have not been used. The files that have been used most recently are kept on disk, whereas files that have not been used for the longest period are replaced and migrated to tape.

The LRU algorithm is a 'reactive' algorithm. That means that it will never migrate data from secondary to primary storage without an (explicit) access by a user.

Chapter 3

Data Mining for a Prefetching Algorithm

3.1 Introduction

The goal of this chapter is motivate the use of data mining techniques in order to improve the performance of the Shell digital archive. The motivation is based on challenging the reactive character of the currently used replacement algorithm, Least Recently Used. Furthermore, achievements in former researches provide the motivation for replacing the algorithm with a pro-active one. The concept is called prefetching, meaning that the algorithm fetches files to store on primary storage before (therefore 'pre-') a user has requested them.

The chapter starts in Section 3.2 by challenging the Least Recently Used algorithm to be used in the current situation. Subsequently, in Section 3.3 related literature is used to motivate the research towards a prefetching algorithm based on data mining results. Furthermore, it states what this research contributes to former researches. Section 3.4 introduces the research questions. Section 3.5 describes which methodologies are used in order to answer these research questions.

3.2 Challenging the Least Recently Used Algorithm

When a user accesses a data set that is present on disk, the time it takes that a user can actually use the dataset is negligible. However, when it is not present on disk, it can take up to multiple hours to read the data from tape.

Performance of a layered storage environment can be measured in terms of hit rate. It is calculated by dividing the number of accesses of files when they are present on primary storage (disk) by the total amount of accesses.

The Shell archive for seismic data is a multi-layered storage environment with a disk as primary and magnetic tapes as secondary storage layer. The currently used replacement policy, the Least Recently Used (LRU) algorithm, keeps the most recently used files on disk and the remaining files are stored on tape[1].

The system is kept separated from active storage environments. Moreover, a user actively puts his data in passive working storage (archive) or copies data from the passive to his active storage environment. The difference between active and passive storage is observed in what a user can do with the data: seismic processing is executed from active storage. We call this situation an 'active archiving' situation.

In a situation of 'active archiving', it is assumed that just archived files will not be used for a long period[2]. Additionally, just de-archived files are transferred immediately to a user's active storage environment to work with the files. Hence, the same files will probably not be accessed soon again. The choice of the LRU algorithm as replacement policy to decide upon the content of the disk does not align with the situation of 'active archiving'.

It is expected that the hit rate of the digital archive of seismic data within Shell can be improved through using another replacement policy. This improved algorithm should consider other elements than (only) the last access of a file. Furthermore, the algorithm could be 'pro-active' instead of 're-active'. That means that it can decide itself to migrate data to primary storage without an explicit access request by a user. The problem remains what criteria should be used to decide upon which files to keep on primary and which on secondary storage.

Furthermore, it must be stated that the implemented algorithm must not constrain the system heavily. That is, although an algorithm may theoretically be far more optimal in terms of hit rate, it may increase system overhead such that the overall performance is decreased. Therefore, the performance is also calculated in algorithm overhead.

3.3 Literature on Replacement Algorithms for Layered Storage

This section describes general literature regarding replacement algorithms. These researches do not consider the situation at Shell, not are the researches executed at Shell.

[1] Replacement policies and the LRU algorithm are introduced and explained in Section 2.3 on page 6.

[2] 'Active archiving' is introduced and explained in Section 2.2.5 on page 5.

The LRU algorithm as a replacement algorithm has been challenged by other researchers before. According to the author in [15], the standard LRU algorithm is static. His field of research is layered storage in general. The standard LRU always considers the same amount of files to be migrated. However, the author claims that this constrains the system, as the replacement algorithm takes a standard overhead cost constantly. When traffic on the system increases, this situation constrains the response time. Hence, he concludes that when there is more traffic, the replacement algorithm must be lowered in activity. The research considers the number of file replacements as a variable and dependent on the traffic. Due to this adaptation, the overhead of the system decreases. The author uses a numerical, theoretical simulation to show that the overhead of the dynamic LRU can become to a third of overhead of the static LRU. A lack of this research is that it does not test the assumptions in a real situation. Furthermore, it does not take into account the sizes of the different storage layers. Therefore, if traffic is heavy and will stay heavy for a long period, the total system will use more storage than the maximum capacity. Additionally, they conclude that only access history is used to predict file demand, whereas other factors may be likely to predict file demand as well.

The authors in [16] also adapt the LRU policy. Their field of research is Web Cache Management. Web Cache Management focusses on which (parts of) web pages should be stored in which layer of web-servers. All parts must pass the primary storage to reach a user. This situation is, thus, comparable to a layered digital archive. In their research, they stat that the standard LRU only considers the *last* access of a file. They add components for former accesses. They use exponential smoothing to estimate each file's mean time between two accesses. Exponential smoothing recursively takes into account as many accesses as observed, whereas the last accesses are most prominent in the estimation of the mean time between two access requests. They improve the hit rate in several experiments by 5% to 10%. A strength of this approach is that files are all considered as unique with unique access statistics. However, this research still lacks to incorporate other factors than access requests.

Although both researches show an increase in performance through using an extension to the LRU algorithm, they still consider a file's access history to be predictive for a file's future accesses. They both do not consider other factors than time. Furthermore, these algorithms are still 're-active'. In the research in [16], the adaptation to the LRU does assure that the length files remain on primary storage differs for separate files. However, pro-active migration does not take place.

In [27], the concept of prefetching is introduced. Prefetching is a pro-active replacement policy. Prefetching means that the algorithm can determine to migrate data from secondary to primary storage. The research focusses on hierarchical storage models. The attractiveness of files is determined by means of a genetic algorithm. They establish a fitness function and regularly update

the attractiveness values of files. They achieve a performance improvement of 42% in terms of hit rate.

In [3], the authors extend this prefetching algorithm. Besides general attractiveness as in [27], they use data mining techniques to construct web access models. Through mining web access logs, they are able to construct a decision tree and to find frequently observed patterns of file accesses. They show that both these approaches enhance the performance in terms of hit rate compared to the LRU algorithm. However, they state that the strength of using the data mining results is largest when adaptations to the results can be implemented easily. Herewith, they mean that the 'knowledge' they extract from historical access data can change over time. Certain web-pages become (im)popular, either in general or for one person specifically.

The same concept of using data mining for a prefetching algorithm is proposes in [11]. They use a sequence miner for 2-sequences from web log data. The results of the sequence miner is knowledge in the form of rules of the format $A \rightarrow B$. These rules are used to calculate the probabilities of files to be accessed. The files that are most likely to be accessed soon are 'buffered', which is similar to prefetching. Compared to [3], they add a concept that is comparable to the aforementioned concept in [15]. They adapt the size of the buffer when traffic is very heavy, therewith making the algorithm dynamic for traffic changes.

In [2], the idea of prefetching based on data mining results is embraced. However, with their results they show that traffic can increase heavily due to a prefetching algorithm. Therefore, they argue for a trade-off between prefetching and computing overhead. By actively setting system constraints for the prefetching algorithm, one is able to gain from the benefits, while not facing the problems of reducing system performance.

These researches show to successfully challenge the LRU algorithm. Both general likelihoods and access likelihoods due to relations between files are considered. However, they all lack to incorporate generalisation of this 'knowledge' by means of data taxonomy.

3.3.1 Contribution of This Work

The discussed researches in [2], [3], [11] and [27] show the potential of implementing a prefetching algorithm based on data mining results. However, as these researches focus on web-page Cache Management, they do not consider data taxonomy. Hence, knowledge that is extracted by means of data mining is not generalised.

This research uses the concept of pro-actively prefetching data in a different research field: digital archiving. As for both research fields layered storage is used, the replacement algorithms are comparable. As such, digital archiving may benefit from research in Web Cache Management. Moreover, it was argued in Section 3.2 that the LRU is not the most appropriate replacement algorithm.

Hence, the field of digital archiving may benefit from others challenging the LRU for similar reasons.

As only a small percentage of an archive is de-archived (feature of an archive), it is only useful to extract the relations in a generalised format. Therefore, the concept of prefetching based on data mining results is extended by generalisation of extracted knowledge. This concept of generalisation of data mining results has not been implemented in a prefetching policy yet. Specifically, we will test this generalisation on spatial coordinates, which has not been executed before either.

3.4 Research Questions

The research questions are derived from the current situation and former research. These researches show the potential of extracting knowledge from historical access data through data mining. Data mining is very powerful for very large databases, as is also the case for the Shell digital archive.

Additionally, the knowledge is used successfully to pro-actively prefetch data to higher storage layers to improve the hit rate of layered storage systems.

3.4.1 Main Question

How can one improve the performance of a digital archive of seismic data by using a prefetching algorithm as replacement algorithm, based on the results of data mining on historical access data?

3.4.2 Sub-questions

The main question will be answered by answering the following sub-questions:

1. Which data mining techniques are most appropriate to extract useful knowledge from historical access data?
2. What knowledge can be extracted to be used in a prefetching algorithm?
3. What does a prefetching algorithm look like, based on the results of the knowledge extraction process?
4. What is the performance of the prefetching algorithm compared to the currently used Least Recently Used algorithm?

3.5 Methodology

The methodologies that are used in this research differ for each research question. They are discussed for each research question separately.

Sub-Question 1 Data mining shows to be very useful in extracting knowledge from very large databases [13] [28] as is the case for this archive.

Many definitions and application areas of data mining exist [13] [14]. This aligns with the number of available techniques. Former researches show the application of different kinds of techniques, indicating that a choice must be made. Therefore, literature research is executed to give an overview of data mining and its applications, extract selection criteria for this particular research and making the choice for which technique is most appropriate.

Sub-Question 2 The second Sub-Question requires the answer to the first research question as input.

To steer the process of data mining, a common standard is developed by experts from the data mining industry. As this framework is based on results of data mining experts, it is chosen to follow this framework as close as possible. This framework describes the separate phases of a data mining process: business understanding, data understanding, data preparation, modeling, evaluation and deployment. The model is called the **CR**oss **I**ndustry **S**tandard **P**ractice (CRISP) model and is given in Figure 3.1.

Applying the data mining techniques is executed with an open-source tool called Weka [28] [26]. The reason for this tool to be used is that it has implemented (almost) all available data mining algorithms. Using one tool for all data mining processes (if more than one is required) causes the results to be comparable.

Sub-Question 3 The results of the data mining process forms the input for the prefetching algorithm. The algorithm is simulated to test the performance by means of Visual Basic Macros in Excel. Besides the proposed algorithm based on the results of the data mining process, the LRU algorithm is modelled in the same way.

The reason for choosing Visual Basic Macros is that the data is organised in a manner that is easy to handle with a spreadsheet application. Furthermore, the output of the data mining tool is easily converted to the Excel application. The reason for using the same simulation tool for both algorithms is again to make comparison easy.

Sub-Question 4 The performance comparison between the two models that provides the answer to research question 3 is based on two elements, derived from literature 3.3 [2] [15] [18]:

1. the hit rate;

2. algorithm overhead.

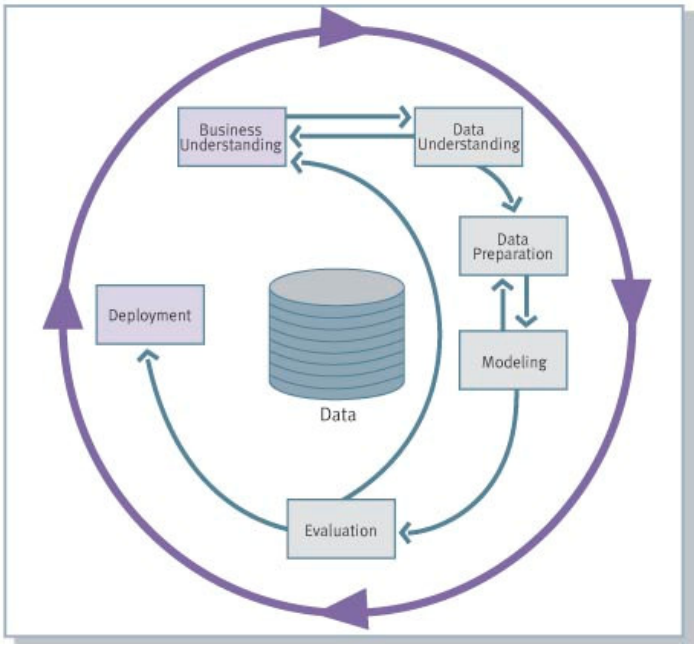

Figure 3.1: Cross Industry Standard Practice Methodology for Data Mining

The hit rate is a predictor of the rate of the system constraining the user in waiting for accessed files. As this can take up to multiple hours, it is useful to improve this ratio.

Although the hit rate is useful, it must not be improved to every price. That is, if the algorithm is too large that the system is constrained in its primary activities (data migration), the algorithm fails.

Chapter 4

Choosing the Data Mining Techniques

4.1 Introduction

The goal of this chapter is to provide insight in data mining in general. Furthermore, the chapter aims at choosing the appropriate data mining techniques to extract knowledge from historical access data of a digital archive of seismic data.

In section 4.2, a general overview of data mining tasks and techniques is given. It continues in section 4.3 by providing the criteria to choose among the different data mining techniques for this particular problem to extract knowledge from accessing seismic data. Section 4.4 then applies the selection criteria and assesses which data mining techniques are most appropriate. Two additional sections 4.5 and 4.6 provide more technical details of the two data mining techniques that are chosen: clustering and association rule mining.

4.2 Overview of Data Mining

A clear definition of the process of data mining does not exist [13]. However, all definitions describe it as a method to prove (sometimes non-obvious) relations between data sets [14]. Furthermore, the definitions claim that the goal is to increase the value of data through a better understanding of these relations [14] [29].

Hence, the definitions consist of an '*action*' (prove relations) and a '*reaction*' part (increase data value). The goal of data mining heavily determines the way the reaction part must be filled in. On its turn, the appropriate action part is derived from the reaction part of the data mining process. However, before knowing what *can* be the goal, it is useful to know which techniques (the content of the action part) are available.

4.2.1 Data Mining Tasks

The following tasks are able to be distinguished by means of using a data mining algorithm [13]:

1. description;
2. estimation;
3. prediction;
4. classification;
5. clustering;
6. association.

When analysing this enumeration of data mining tasks, one can split the tasks (at least) on two variables. The first variable is whether it is an *action* or a *reaction* activity, which determines which phase of the data mining project it belongs to. The former three tasks are reaction tasks, whereas the latter three are action tasks. That means that in the process of data mining, the first three tasks can be executed after first applying one of the latter three tasks.

The second variable to distinguish the action tasks (latter three) further is the level of supervision. Classification requires supervised learning, whereas clustering and association are commonly executed in an unsupervised way. Supervision means that via knowing answers (classes) from historical examples, new examples are dealt with. Supervision requires the input data set to be suitable for it, which is not always the case.

4.2.2 Overview Data Mining Techniques

Classification Techniques A class is a pre-determined area to which examples belong, according to their attributes. Classes are mutually exclusive. Furthermore, examples from the input data set belong to exactly one class. However, after learning classification by examples, it is not definitely the case that *all* possible solutions are classified. Hence, the conjunction of all classes do not capture the whole universe of the solution field.

Classification occurs on the basis of historical examples. So called 'case-based reasoning' is used to classify new examples. Whereas the classical method k-Nearest Neighbor and Decision Trees cannot handle uncertainty and new cases due to its use of strict analogy with historical examples, modern techniques are more sophisticated in that sense. For example, neural networks use the method of backpropagation to adapt weights slightly and as such, can classify all examples, even with much noise.

A disadvantage of classification in general is that the algorithms need many examples to form a good output. Inherent to that fact lies the problem that changes in trends or input are noticed very slowly. The algorithms can adapt on-line (learn during usage) but will always adapt slowly. Another shortfall is the fact that knowledge is sometimes implicitly hidden in the output. Hence, one can 'only' **use** the output, whereas sometimes *understanding* the knowledge would be preferable. A final shortfall is the fact that the classification must be known in advance for each example, so the algorithm learns through supervision and feedback.

Advantages of mainly neural networks are the ability to handle noisy data and the aforementioned advantage of on-line learning. Furthermore, classification is very powerful when classes are known in advance.

Clustering A cluster is an area within the solution field to which part of the examples or instances belong. The number of variables chosen to base clusters on can vary per clustering project. Clusters are characterized by a cluster centroid. New examples are put in a cluster, based on their distance to the cluster centroids. Although a cluster is characterized by its centroid, it is commonly graphed as the group of cluster members. This groups might overlap with each other. Although *each* new example is clustered, the centroids do not capture the whole universe of the solution field per definition.

Clustering and Classification do not differ heavily. The major difference is that clustering does not require the identification of the clusters (or classes in analogy with classification) in advance. This is a benefit, but a shortcoming as well. That is, until now appropriate measures of determining the adjust amount of clusters have not been generated. Several measures exist to identify the 'per cluster value' and as such, one can stop splitting clusters until performance is not increasing enough anymore.

The basis of clustering are to present examples one by one to an algorithm, put them in a cluster and adapt the cluster attributes by means of its centroid. The mostly used method is to minimise the mean squared Euclidian distance. The equation of the Euclidian distance is given in equation 4.1:

$$d_{Euclidian}(x, y) = \sqrt{\sum_i (x_i - y_i)^2} \tag{4.1}$$

Differences in clustering techniques are the rate of supervision and complexity of the algorithm. However, the basis is more or less the same for all. In section 4.5, more details are given regarding clustering.

Association Association between data sets means they will jointly act in certain transactions. A famous example is the fact that on Saturdays beer and diapers are commonly acquired jointly in supermarkets. It seems that working

men go to the supermarkets in the weekend and buy themselves a nice beer, next to the other things for the household. Hence, supermarkets can arrange their products, such that they use that knowledge to gain benefits from it. In this example, beer and diapers are 'associated'. Association rules are always in a format like $A \rightarrow B$, in which A and B can be either one or more predicates, separated by conjunctions or disjunctions.

Association rules find their use in very large databases, for the reason that the algorithms are light and rules are found easily. Furthermore, the output of association rules is very easy to interpret and can, thus, be used in a later stadium of a (data mining) process.

A disadvantage of association is that it commonly requires large item sets. Large item sets are associated items that occur in a large percentage of all transactions. This means that rare rules will stay unnoticed, whereas such rules can have high value. Another disadvantage of common association rule mining projects is the lack of ability to 'understand' the knowledge. In other words, if A and B associate, it is sometimes more important to know *why* they associate. Several researchers challenge this issue with generalised association rules (discussed in Section 4.6 on page 24). Furthermore, association rule mining is often characterized by the generation of many useless rules. This is caused by the nature of using metrics. A more detailed explanation can be found later in this chapter in section 4.6.1.

4.3 Selection Criteria

Now that we have learnt about which techniques and tasks are available, it is time to look at which criteria determine the technique that is used to extract knowledge from the historical access data of seismic data.

Firstly, the goal must be formulated. The eventual goal is a proposal for a prefetching algorithm. Challenging the LRU algorithm as discussed in Chapter 3 leads to the following two statements:

1. (only) time-based algorithms like the LRU algorithm are not appropriate;
2. (sequences of) de-acrhiving/access activities do not take place randomly.

The first item, combined with the dual definition of performance measures (hit rate and overhead), drives the first selection criterion: algorithm overhead. The algorithm must be as 'light' as possible. Although the LRU is challenged and may be replaced, it cannot be replaced to every price. The algorithm overhead cannot be too complex.

The second item in this enumeration leads to more criteria. First of all, it challenges random accessing. This can mean two things, also illustrated in Section 3.3 on page 10 in former researches. Firstly, some data sets are more

attractive than others in general. The authors in [27], for example, implement this concept by a genetic algorithm. Secondly, relations between data sets can determine the sequence of accesses. So, the access of data set A can increase the likelihood of accessing data set B soon. This is shown in [3] and [11].

Generally, a (more optimal) replacement algorithm also contains a criteria of probability. That is, the algorithm must try to predict the future to put the most optimal required files on primary storage.

Furthermore, knowledge that is extracted by means of data mining must be easily understandable as to incorporate in a prefetching algorithm. That means that knowledge cannot be hidden in the algorithm or the output.

A final criteria is that (part of) the data mining process is executed in an unsupervised way. The reason is that a judgement about the existence of the relations that are sought is impossible. It is not known in advance whether relations exist.

Criteria that are not required in this project, but have been considered are:

1. the need for an executable program;
2. on-line learning;

4.4 Assessment

In Table 4.4 one can see the results of assessing the techniques to the criteria mentioned in the previous section. These results are based on the findings from the sources [13], [14], [28] and [29]

		Overh	Gen.Prob.	Seq	Prob	Knowl	Noise	Unsup
Classif	classic	++	+	−	−	+	−−	−−
	ANN	++	+	+	+	−−	+	−
Clust		++	++	−−	++	++	+	++
Ass		++	−	+	+	++	+	++

Table 4.1: Assessment of data mining tasks

ANN is an abbreviation for Artificial Neural Networks.

From Table 4.4, one can conclude that the three tasks all have benefits and shortcomings regarding the selection criteria. It is clear that classification is not appropriate, as the level of supervision is too large: the determination of classes in advance is impossible.

Clustering lacks the opportunity to incorporate sequence relations. Therefore, only executing clustering is not sufficient.

Therefore, it is chosen to use both clustering (for static/general attractiveness) and association rules (for dynamic/interrelated data set probability). This

combination fulfils the statements as stated in the enumeration of the selection criteria.

In the next two sections 4.5 and 4.6, more detailed descriptions are given of these two techniques.

4.5 Clustering

4.5.1 Heuristic Clustering

The simplest clustering algorithm takes as input the number of clusters and first, randomly assigns initial values to these clusters. Subesequently, the separate training points are offered and assigned to the closest cluster. When adding points to clusters, a new cluster value is calculated by minimizing the squared distance of all cluster members to the cluster centroid.

Although this method is very powerful, an important choice remains up to the user: the amount of clusters. One way of dealing with this issue is to iteratively increase the amount of clusters and check each time the performance of each cluster. A common way of measuring the overall quality is the Category Utility function [28], of which the equation is given in equation 4.2:

$$CU(C_1, C_2, \ldots, C_k) = \frac{\sum_l Pr[C_l] \sum_i \sum_j (Pr[a_i = v_{ij}|C_l]^2 - Pr[a_i = v_{ij}]^2)}{k} \tag{4.2}$$

$\ldots C_1$, $C_2, \ldots, C_k$ being the k clusters, a_i the attribute which takes on values v_{i1}, v_{i2}, $\ldots$ which are summed over j, the members of the particular clusters. The probability on its turn is summed over all clusters l (running from 1 to k). This expression compares the probability of an example having value v if the example belongs to a cluster C_l to the general probability of the same example having the value v. If this difference is not big, it does not make sense to include the instance as member of the particular cluster.

Dividing by k in equation 4.2 is done to prevent 'overfitting'. When dividing by k would not be applied, the maximum category utility would be reached when all instances form their own cluster.

Equation 4.2 in this exact format, describing the Category Utility of clustering, can only be applied to nominal attributes in this form, although it can be extended by including a normal distribution of the values of the instances.

4.5.2 Probability-based Clustering

The heuristic approach to clustering faces several shortcomings, of which the most critical one for this research is the arbitrary choice of the amount of clusters in advance.

Another approach to clustering is probability-based clustering. This appraoch defines clusters as probability distributions to which members belong

to a certain extent. The distributions (of clusters) all have their own mean μ and standard deviation σ. Furthermore, the prior probability p_X to belong to a cluster X adds up to 1. For example, if two clusters would be present (A and B), the prior probability of an instance to belong to cluster A (p_A) plus the prior probability to belong to cluster B (p_B) adds up to 1, while p_A and p_B may be of different value.

A mixture model can be constructed with all clusters having a μ, σ and a p, which can be calculated by the distribution of instances between the clusters.

Problems occur with the forming of these clusters because they must follow from the data itself instead of determination in advance. The calculation of the different values (different clusters) for μ, σ and p is impossible to execute in advance. Therefore, the Expectation Maximisation algorithm iterates the calculation of these values. It first 'guesses' the values of all the parameters, starting with one single cluster. Subsequently, it uses these parameters to calculate the probabilities that instances belong to clusters and re-estimates the parameters after having offered new examples. Furtehrmore, it is checked whether splitting the cluster(s) to increase the cluster utilities and general score of the cluster model.

In this way, the content of the clusters follows from the data itself, which is the goal of unsupervised learning. The equations belonging to the parameter estimations are given below in Equation 4.3 and 4.4.

$$\mu_A = \frac{w_1 x_1 + w_2 x_2 + \ldots + w_n x_n}{w_1 + w_2 + \ldots + w_n} \tag{4.3}$$

$$\sigma_A^2 = \frac{w_1 (x_1 - \mu)^2 + w_2 (x_2 - \mu)^2 + \ldots + w_n (x_n - \mu)^2}{w_1 + w_2 + \ldots w_n} \tag{4.4}$$

$\ldots x_i$ being *all* the instances instead of those belonging only to cluster A. w_i is the probability that instance i belongs to cluster A.

One of the disadvantages of the EM algorithm is that the ending point is hard to determine. The nature of the algorithm shows that it will converge to a fixed point but in practice this point will never be reached. So, one should arbitrarily stop the algorithm when the increase in performance is negligible. The performance is given in multiplying conditional probabilities as in equation 4.5. This equation calculates whether the clusters follow from the data set, or that they still reflect the initial 'guesses'. The equation calculates the conditional probabilities of examples being part of the clusters and multiplies the addition of the cluster probabilities per example. Although the multiplication part eliminates the properties of real 'probability', this equation is still a good measure to compare cluster distributions with each other.

$$\Pi_i (p_A Pr[x_i|A] + p_B Pr[x_i|B]) \tag{4.5}$$

... for a result with two clusters A and B, both with a normal distribution $f(x, \mu, \sigma)$.

In practice, the clustering is iterated until the difference between successive values is less than a certain number for a certain number of successive iterations.

Another shortcoming of the EM algorithm is that the maximum it finds might be a local maximum. This can be avoided by running the algorithm multiple times in different orders of presenting the examples.

4.6 Association Rule Mining

Association rule mining is heavily used in finding correlations between data sets in a large set of transactions. Each transaction contains x items with $0 \leq x$. Association mining will find the Y best rules in the form $A \rightarrow B$, with Y being a pre-determined number. Sometimes this number is not reached. That means that not enough 'large item sets' are found due to minimum requirements for the used metrics.

The most classic but still heavily used algorithm is the Apriori algorithm. This algorithm focusses on the identification of large item sets. Large item sets are found by determining minimum metrics in advance. The most common metrics are Confidence (rate of rightness of a rule), Support (rate of occurrence) and Lift (Ratio of experienced Confidence to expected Confidence). When having set the metrics, a large item set is a set of instances within one transaction that fulfils the metrics.

After having identified large item sets, the item sets are combined and split in order to find the most appropriate rules. The underlying assumption that steers this process is that if X is a large item set of size k and $Y \subset X$ of size $k-1$ that Y *must* be a large item set as well. As such, the algorithm continues to build the most relevant sets by exploring the large item sets one by one.

4.6.1 Benefits & Shortcomings

A large benefit is the straightforward, low resource consuming algorithm. Secondly, the outcome of the process of association rule mining is easily understandable and easy to use. The metrics confidence and support might be used in establishing conditional probabilities between items and as such, one can establish Bayesian networks [7] [19] and/or Markov chain modeling [4] [8] [9] [29]. Bayesian Networks are networks in which conditional probabilities between variables are represented. Walking through these networks can be used to identify what ill happen with new data. Markov Chain modeling uses the conditional probabilities to define possible states and the probability to transfer between these new states. These two methods are very useful when a large percentage of the data is used. Otherwise, they are very heavy regarding computing overhead.

Problematic is the generation of a large number of useless association rules. Especially when one is interested in strong, but not so frequent occurring associations, association mining algorithms will create much noise [13] [14] [28]. That means that human intervention both in advance and after the process of rule generation is required to appropriately use the results.

A second major shortcoming of association rules is that it is hard to automatically incorporate a hierarchical format of data. That means that the algorithms are able to find associations between files on the same data level, but fail to 'understand' the data well enough to formulate generalised rules. This, however, is important in this research. Again, human intervention is required to overcome this shortcoming. The concept of generalized association rules is described in [10], [22], [24] and [25]. The most common way of generating generalised association rules is to manually iterate the association rule finding processes on different levels of the taxonomy[1].

The taxonomy of seismic is determined through interviews with domain experts. Furthermore, the research database reflects this data taxonomy. This database is described and discussed in the next chapter (describing the Experimental Set-up of Data Mining) and in more detail in Appendix A.

[1] Taxonomy stands for the hierarchical relations of data, e.g. a tuna *isa* fish *isa* animal.

Chapter 5

Experimental Set-up to Extract Knowledge

5.1 Introduction

The goal of this chapter is to define the details of the experiments that aim to extract useful knowledge from historical access data. Two different kinds of relations are analysed: relative (static) attractiveness and joint accesses of files.

The process of data mining follows the CRISP methodology (see Figure 3.1 on page 15). During the Business Understanding phase, three experiments are identified. They all assess one or more hypotheses to extract knowledge from the accesses of seismic data.

Before the modeling and evaluation of the data (both by means of data mining), the data must be understood and prepared (see again Figure 3.1 on page 15). This process results in a database, which is especially created for this research and is based on data organisation and management within Shell. Details and considerations regarding this database are given in Section 5.2. The three Sections 5.3, 5.4 and 5.5 that follow all describe in more detail one experiment each. These sections complete the 'Data Understanding and Preparation phase' and are the input for the 'Modeling' phase, of which the results are given in the next Chapter 6.

5.2 Database Design

The database design follows the methodology as proposed in [6]. The starting point is the creation of an Entity-Relationship Diagram (ER Diagram). This is developed by means of interviews with domain experts. Based on the ER Diagram, the database tables are constructed and normalised. One of the main requirements that is worthwhile to notice is the element of spatial objects. It will be assessed whether certain data sets are (commonly) accessed jointly by means of association rule mining. In section 4.6 it is also explained that rules are

to be generalised where possible. One of the variables on which generalisation might be useful is spatial distance. Hence, the database must be capable of executing spatial queries to test the hypotheses and rationales.

The choices and the meaning of the concepts is only briefly described in this chapter. For a more detailed description of the relevant choices regarding the database design, we refer to Appendix A in which more about the data organization within Shell is explained.

5.2.1 Features and General set-up

The database is derived from the data organization within Shell. The content of this database is extracted from four separate databases. The main database is the archive itself. This contains the historical access data and of course the seismic data itself. However, to understand the data and to incorporate a spatial elements, data from other databases is extracted as well.

No formal connection between any of the four databases within Shell exist. Therefore, not all data in the archive can be used. In terms of size, the research database contains around 20% of the metadata of all available data, the value of ratio depending on which database is regarded as the basic one.

5.2.2 Entity-Relationships Model

Figure 5.1 on page 29 shows the result of forming the Entity-Relationship Diagram. The most important entities and relations are:

1. a seismic survey (entity) is a polygon shaped object with coordinates
2. data from a survey of seismic data is collected in one or more directories (entity) on the file system of the archive;
3. a directory can contain data from one or more surveys (more surveys in case data from different surveys is merged);
4. a directory contains several files (entity), which can be either of the same type or different types;
5. files can be archived (entity) or accessed (entity);

5.2.3 Database Tables

From the ER Model, a relational database scheme is derived. This scheme is given in Figure 5.2 on page 30. The database contains six tables. For each entity in the ER model, a table is constructed. The entities are illustrated in the enumeration in the previous section. Furthermore, the M-N relation between survey and directory forms an additional table.

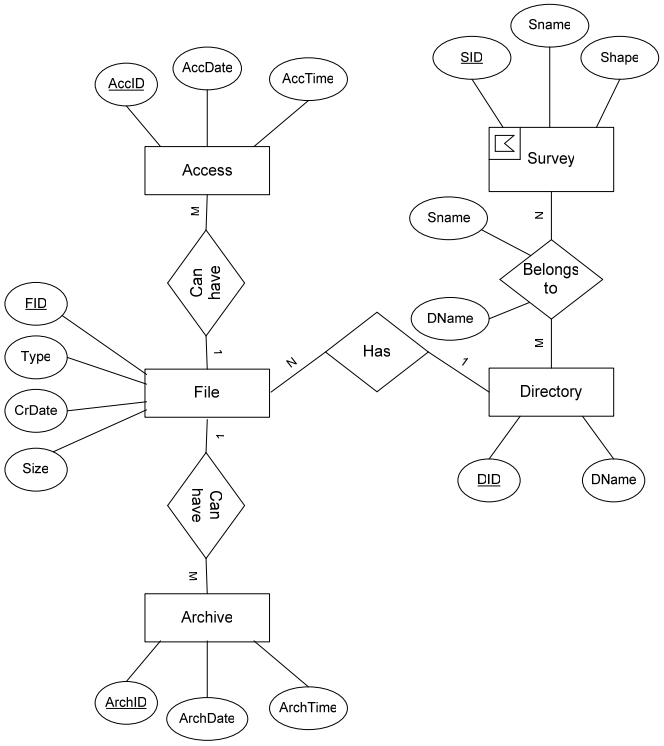

Figure 5.1: Database ER Model

The tables 'Archive' and 'Access' are connected to table 'File' via the FileID. The table 'File' is connected to table 'Directory' through the DirectoryID. The M-N relation, table DirBelongSurv is connected to both the tables 'Directory' and 'Survey', respectively on DirectoryID and SurveyID.

5.2.4 Database Numbers

Some numbers are given of the content of the database. The database contains:

1. 158 surveys;
2. 253 directories;
3. 259,632 files, created between 1998 and 2007 with a total size of approximately 85 TeraBytes;

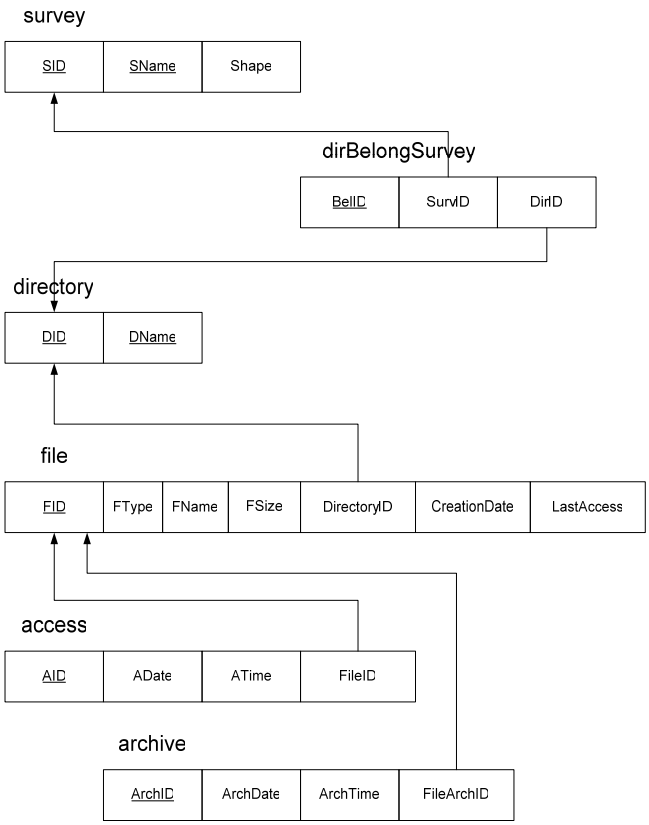

Figure 5.2: Database Relational Scheme

4. 50,749 accesses, varying from March 1st 2006 to March 31st 2007 (13 months);

5. 17,794 archive requests from the same period as the accesses.

It is chosen that the training set consists of all relevant data until January 1st 2007. The data as from January 1st 2007 until March 31st 2007 is used as the test data set for a simulation of the prefetching algorithm and the LRU algorithm.

5.3 Experiment 1: Decreasing Static Likelihood

Experiment 1 assesses two hypotheses. Both hypotheses state that the likelihood of certain files decreases by the presence of other files in the archive. Two attributes of files are assesed: firstly the type of file and in particular, files of type 'raw' and type 'idented' and secondly, the 'oldness' of files (creation dates).

The results of Experiment 1 can be used in a prefetching algorithm by giving a very low value in terms of likelihood to certain (groups of) files and thereby, relatively increasing the value of other files. These hypotheses are both derived from interviews with domain experts (seismic processing geophysicists) within Shell.

Hypothesis 1: *If and only if within a survey A files of type 'raw' and files of type 'idented' exist and the idented files are not too old, the idented files will be more likely to be accessed than the raw files*

Rationale The rationale underlying this hypothesis has already been briefly described in section 2.2.2. Identing is required for *every* processing project. So, for each project raw files must be tranformed into idented files if idented files are not present. Idented data is then used for further processing activities. Therefore, it is imaginable that an idented data set is more attractive to access than a raw one. That is, the project could be finished faster with comparable results.

Method of Assessment This first hypothesis is fairly straightforward and does not require unsupervised machine learning techniques. Therefore, the method of assessment is simply counting the number of times this hypothesis is true. As there exists 'fuzzyness' with respect to the oldness of the files, we distinguish the data on the creation dates of idented files. It is chosen to do so on a yearly base.

Furthermore, it is required to normalise the data to the availability of idented files. The investigated years differ to a large extent regarding the amount of created idented files. This is highlighted in Figure 5.3. In this Figure, we see that in 1998 many more idented files were created than in the other investigated years. Therefore, the data is normalised to eliminate strange distributions.

Hence, the assessment of this hypothesis consists of:

1. searching for the amount of accesses of files of type 'raw' when within the same survey files of type 'idented' are present;

2. Record the creation date of the files of type 'idented' and divide the number of raw accessed files over the years of created idented files;

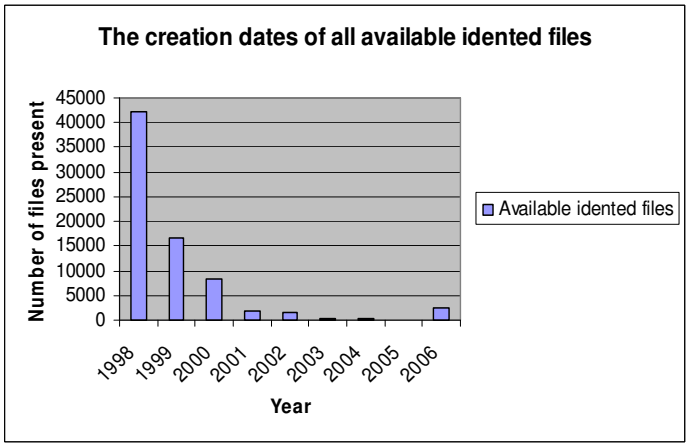

Figure 5.3: Number of available idented files per year

3. Identify the number of times files of type 'idented' are accessed and divide them over the years of creation dates.

Subsequently, a graph is constructed that represents the 'attractiveness' of idented files for each year of this research (1998-2006) compared to the 'attractiveness' of raw files. The raw and idented files together form the whole set (100%) of accesses, whereas the intersection between the accesses of raw and idented files is the empty set.

The expectation is that during the first years, there will be a random distribution of attractiveness of idented and raw files. Towards more recent years, it is expected that the relative attractiveness of idented files increases compared to raw files. The goal is to look for the moment from which idented files are more likely to be requested than raw files.

Hypothesis 2: *When one accesses a data set of type A from survey B, it is likely that one accesses the most recent data set of type A within the same survey B*

Rationale The rationale underlying Hypothesis 2 is the evolution of the IT world. Seismic processing is executed by using heavy algorithms, which still do not lie within the capabilities of IT equipment. Therefore, abbreviations and approximations are developed by research geophysicists to incorporate the best possible, executable algorithm. However, CPU power and digital storage increase at a fast pace. So, there exists a short business cycle of using the algorithm before adaptations to the algorithms are ready to be used.

Therefore, although two data sets are of the same type (processing phase), it is assumed that the most recently created version delivers the best result (best image, more noise reduction etc.)

Method of Assessment This second hypothesis of Experiment 1 is also fairly straightforward. Actually, the hypothesis is a rule, which must be confirmed or falsified by evidence from the experiment. Therefore, a table is constructed that shows the rightness of the rule. The metric that is used is Confidence. This metric is heavily used for association rules (see Chapter 4, Section 4.2 on page 17). Confidence is a rate of success of the rule. It counts the number of successes and divides it by the addition of successes and failures. A failure is only counted as a validation of the antecedent of a rule (A in $A \rightarrow B$) and a negative value for the consequent of the rule (B).

Although the method of proof may seem straightforward, the assessment of rightness of the hypothesis is not. That is, the Confidence as a metric is easy to find, but there is no strict rule which states that the hypothesis is confirmed through a Confidence value of above a certain threshold $X\%$. Therefore, after having constructed the table, one can only conclude the confirmation of the hypothesis to a certain level. However, although we may not have a hard conclusion about this hypothesis, the outcome can be used in a caching and replacement algorithm. This is further explained in Chapter 6, Section 6.3 starting on page 43.

Furthermore, the concept of 'data set' is somehow complicated. From the data itself, it is not tractable which files exactly belong to each other as one data set. Hence, it is not useful to only look at the *most* recently created file. Therefore, it chosen through interviews with users, that a period of three months is taken into account. A success for the rule is counted when file X of type A is accessed and no file Y of type A is available in the archive, created more than three months later than file X.

5.4 Experiment 2: Relative Static Likelihood

Hypothesis *If file A and file B have different attributes, their general static likelihood to be accessed differs.*

Rationale This hypothesis basically states that accessing files from the archive is not executed randomly. Besides that, it states that one can measure the general attractiveness of a file by means of its attributes.

The attributes are chosen through interviews with domain experts and are the:

1. type of file;

2. directory a file belongs to;

3. survey a file belongs to;

Many different file types exist. The file type indicates which processing algorithm was used and what the file can be used for. All existing file types can be found in Appendix B

The rationale behind this hypothesis is that certain data may be simply more attractive than others. For example, a certain data set is available but the conclusion after a processing project was that no oil nor gas is available within that specific region. The data set will remain in the archive but will not be researched (accessed) anymore soon. Another example is data from a certain type that is accessed heavily for interpretation of the subsurface during an interpretation project.

Specifically, the type of data is important, as this indicates the processing phase and what the data can be used for. Furthermore, the location is important due to the maturity of oil and natural gas fields. This is captured by both the attributes directory and the survey. The separate use of the attribute directory indicates that there may be differences in attractiveness of directories, both in general and per survey (caused by the M-N relation between survey and directory as seen in Section 5.2).

Method of Assessment To assess this hypothesis a clustering method is used. The clustering method that is chosen is the Expectation Maximisation (EM) algorithm. The main reason is that the EM algorithm can be executed in an unsupervised manner. That means that choosing the amount of clusters in advance is not required. It also means that if the algorithm finds more than one cluster, the hypothesis is confirmed.

For all three attributes, a list of the normalised frequency is constructed. These lists separately form the input for the EM clustering algorithm. Hence, the clustering algorithm is executed to find separate clusters three times.

5.5 Experiment 3: Dynamic Likelihood

For an analysis of the sequence of accesses, it is chosen to use association between files (see Section 4.4 on page 21 for the choice criteria and the assessment thereof). The definition as used here for association is:

Definition B is associated with A if the rule $A \rightarrow B$ is extracted from historical access data

Hypothesis *If file A is accessed on time t, the likelihood of accessing file B during the period $(t + \tau)$ increases if and only if file B is associated with file A, τ being small*

Method of Assessment The proof of this hypothesis consists of several phases. In general, proof is sought for files to be accessed during the same period (transaction). If that is the case, one can use these joint accesses by decreasing the time of the period in the replacement algorithm and react upon accesses of (certain) files. Joint accesses in the form $A \rightarrow B$ are constructed to represent association rules. The content of A and B have the following requirements:

1. it consists of one or more predicates, the predicates being a value of the attribute;
2. the predicates are combined by means of conjunction;
3. no negations are present.

Data Collection and First Iteration Data is collected of all transactions. The time period that is chosen is 1 hour. This period length assures the total investigated time to be divided in a sufficient amount of periods (around 7,200 periods/transactions) without having too few relations. This is tested by adjusting the parameter of the period several times.

For all (combinations of) attributes, a 1 is written when accessed during the period and a 0 if not. The tests show that for many transactions, only zeros are present. Furthermore, even when these transactions are neglected, most transactions are dominated by zeros. This results in a large amount of useless rules when considering the goal of seeking rules for joint accesses. Therefore, it is chosen to only consider positive relations and hence, positive instances within transactions. The zeros are filtered from all transactions.

Besides the large amount of zeros, the transactions differ with respect to the amount of positive instances. By testing the maximum amount of positive instances within a rule, it is chosen to maximally consider five instances per transaction. That means that transactions with more than five positive instances are divided through creating transactions with all possible combinations of x instances with x running from 2 to 5.

Aforementioned splitting and mutating of the input data set must be corrected for afterwards. The actual forming of the rules (which is based on metrics like Confidence, Support and Lift) is considered only as *rule generation*, because the metrics are not valid anymore. The metrics of the generated rules are corrected for afterwards by comparing them with the original input data set.

The data mining technique that is used for this experiment is the Apriori association mining algorithm. This algorithm is fairly low in complexity and can handle large data sets easily. Furthermore, the algorithm is 'only' used as rule generator, whereas other algorithms are of high value when the metrics thresholds are determined appropriately.

Due to the fact that the algorithm is only used for rule generation, the parameters are chosen such that the minimum metrics are very low and the number of rules is very large. This causes many potentially useless rules, but they will be eliminated in the correction process afterwards.

The rules are formed on different levels of the data taxonomy: directories, surveys, the combination of directory and file type and the combination of survey and file type.

Significance It has been described in what way rules are formulated. In the process of finding rules, the final step is to check whether the rules are statistically significant. For this process, the expected value for all (combinations of) attributes is compared to the metric of Support multiplied by the metric Confidence of the rule.

An example of calculating the expected value of the rule $A \rightarrow B$ is to multiply the expected values of A and B. As such, one assumes data set A and data set B to be independent and accesses distributed randomly. If this expected value is less than the support multiplied by the confidence of the rule $A \rightarrow B$, then it is assumed that A and B are not independent and the rule is statistically significant.

To eliminate coincidence, a confidence level of 99,85% is chosen. Hence, for the experienced value to be higher than the expected value, it can be stated that it is true with 99,85% certainty. For this confidence interval, the equation for the expected value becomes as in Equation 5.1:

$$E(X) = \overline{x} + 3\sigma \tag{5.1}$$

Chapter 6

Extracted Knowledge for the Prefetching Algorithm

6.1 Introduction

The goal of this chapter is to use the extracted knowledge from the data mining process in a prefetching algorithm. Furthermore, it aims at comparing the prefetching algorithm with the currently used replacement algorithm.

The chapter starts with the results of the data mining process in Section 6.2. The separate experiments are described and discussed on their implications for the remainder of the research. The results as described in these subsections form the answer to Sub-Question 2. Section 6.3 then uses the results to formulate a proposal for the prefetching algorithm DMP. This section forms the answer to Sub-Question 3. The final section of this chapter, section 6.4 compares the results of simulating both the DMP algorithm and the LRU algorithm. This section is the answer to Sub-Question 4.

6.2 Data Mining Results

In this section, the results of the data mining process are given in the next three sections, which all contain the results of one experiment each. Furthermore, for each hypothesis it is stated what the results can mean for the prefetching algorithm. Besides that, the results are discussed at each hypothesis.

6.2.1 Experiment 1: Decreasing Static Likelihood

Hypothesis 1 Figure 6.1 represents the results of the first hypothesis of Experiment 1. It shows the distribution of accesses of raw and idented files, normalised on the creation date of the idented files.

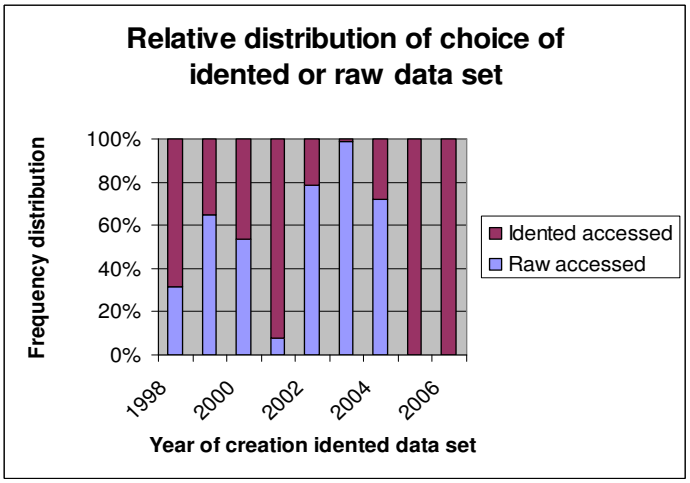

Figure 6.1: Distribution of accesses raw files compared to idented files within the same survey

Several striking elements are visible in Figure 6.1. Firstly, the expectation that recently created idented files are preferable over raw files seems to be true. The idented files that were created during the last two years are more attractive than the raw files for the two most recent year in this research (2005 and 2006).

The second striking element is that the expectation of (slightly) decreasing attractiveness of raw files does not hold. However, the goal was to look for a certain threshold from which point idented files are more likely. Therefore, one can say the hypothesis is confirmed by the results of this experiment. Furthermore, the threshold of two years can be used for the remainder of this research.

Discussion Hypothesis 1 First of all, Figure 5.3 in Section 5.3 on page 32 indicates the very few idented files that were created in 2005. This makes 2005 an outlier and one could argue that the threshold is one year instead of two years. However, as the proposed algorithm is tested with the first months of 2007, the year 2005 will still be taken into account as a year of attractive idented files. However, one must be cautious to use the two years as a standard number in the future.

Furthermore, the expectations of what should have happened before the threshold, are not confirmed by Figure 6.1. This indicates that not only creation time influences attractiveness of files.

The following explanation might be viable. Shell hires third party contractor

companies to execute (part) of seismic processing activities. Especially standard, time-consuming activities like identing are outsourced. However, Shell contains contracts for a certain amount of years, from which point a new contractor company is hired. It could be the case that certain contractor companies are not trusted when their work proves to be insecure. As such, identing results from a certain period might not be trusted at all. When that is the case, the identing process is executed again rather than using the results that are already present in the archive.

The information about the contractor agents is not considered in this research, because the appropriate data is not available. However, this might have changed the outcome of this experiment.

Hypothesis 2 Table 6.1 on page 40 is the result of the experiment to test Hypothesis 2 of Experiment 1. The first column represents all available data types. It consists of codes of data types, of which the meaning is given in Appendix B.

The general rate to which the hypothesis is confirmed is 50% (first row in Table 6.2.1). Although this may seem not a very high percentage, it does indicate that there exists preference to access the most recently created file. The assumption for this reasoning is that more than two data sets exist for each data type within each survey. From all available data sets, in 50% of the cases the most recently created is accessed.

As already stated in Chapter 5, Section 5.3, it is not so important to either confirm or falsify the hypothesis. However, the numbers indicated in the fifth row indicate the *rate* of confirmation. These numbers are used in the remainder of this research to formulate the prefetching algorithm.

Striking to see is that the best confirmation of the rule is noticed for data types with a fairly low Support [1]. This is also shown in Figure 6.2, which shows a scatter plot with on the axes the Confidence and Support. The Figure means that the strongest examples of the rule are noticed only rarely. That means that the rule might have little value in the prefetching algorithm.

Discussion Hypothesis 2 Firstly, the search has been executed for data sets of a certain type. A data set consists of several files. However, which files together form a data set does not follow from the data type itself. Therefore, the experiment has been executed on file level rather than data set level. This means a correction has been made for the creation dates. This was already explained in Section 5.3, starting on page 31.

Secondly, the conclusion states that there exists a preference for more recent data sets, based on the fact that around 50% of the cases shows the access of

[1] Support is the ratio of transaction with a hit (success or failure) of a rule to the total amount of transactions

Type	All	Success	Confidence(%)	Support(%)
ALL	44986	22671	50	50.40
ACIMIG	20	20	100	0.04
ACQPD	5	0	0	0.00
FLDIDT	13432	3880	29	8.63
FLDRAW	12501	7310	58	16.25
GAT	2	2	100	0.00
GATPRO	8265	3360	41	7.47
GATPSI	80	0	0	0.00
MAP	6	6	100	0.01
MIG	848	825	97	1.83
MIGDEP	36	36	100	0.08
MIGDMO	370	340	92	0.76
MIGDTM	44	44	100	0.10
MIGGAT	1303	1299	100	2.89
MIGNMO	70	58	83	0.13
MIGPSI	50	25	50	0.06
MIGZER	122	114	93	0.25
MISCEL	110	81	74	0.18
POSSDM	936	749	80	1.67
PRESDM	4328	2302	53	5.12
PRETIM	1408	1184	84	2.63
PROREP	3	2	67	0.00
STK	253	253	100	0.56
STKDMO	269	269	100	0.60
STKFLT	29	29	100	0.06
STKNMO	319	309	97	0.69
STKUNF	13	13	100	0.03
STKZER	75	72	96	0.16
STTFLD	2	2	100	0.00
VEL	9	9	100	0.02
VELMIG	10	10	100	0.02
VELMOD	4	4	100	0.01
VELPRM	9	9	100	0.02
VELSTK	55	55	100	0.12

Table 6.1: Results of the attractiveness of files through recency

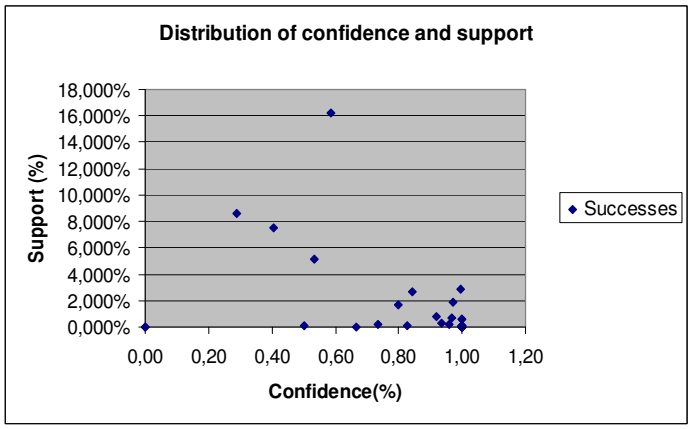

Figure 6.2: Scatter plot of Confidence and Support

the most recent data set, while several older versions exist. However, this assumption of the existence of older data sets is not checked in this research. This lies in the aforementioned problem that it is not completely clear which files together form data sets. One can get an idea about the correctness of that assumption when looking at the table to the amount of files (column 2 of Table 6.2.1). However, how many files belong to one data set and whether that is the same for all data types still remains unclear.

General Usage Experiment 1 Generally, Experiment 1 shows that certain files have a low probability to be accessed due to the presence of other files. Therefore, these low probability files can have a low value and therewith, the other files get a (relative) high value. The results can, thus, be used for the general (relative) attractiveness of files.

6.2.2 Experiment 2: General Static Likelihood

It is checked for three attributes whether clusters can be formed of the normalised access frequency. For all three attributes, more than one cluster has been found by the Expectation Maximisation algorithm. Hence, the hypothesis is confirmed: the attractiveness of files is not randomly assigned, but can be deduced by its attributes. Tables 6.2, 6.3 and 6.4 show the mean, standard deviation and the relative size of each cluster. The mean is be used for the general attractiveness for groups of files. Experiment 2 adds to Experiment 1 other file attributes that determine the general attractiveness of files.

Cluster	Mean	St. Dev.	Inst.	Perc. (%)
0	0.0375	0.0592	19	42
1	1.4317	0.8765	26	28

Table 6.2: Clusters of Attribute Data Type

Cluster	Mean	St. Dev.	Inst.	Perc. (%)
0	0.0291	0.0316	85	34
1	0.3471	0.2221	100	40
2	1.0298	0.1019	27	27
3	1.9941	0.3872	34	13
4	3.4224	0.3438	7	3

Table 6.3: Clusters of Attribute Directory

Discussion The EM algorithm has found more than cluster for all three attributes. In each separate experiment, only the normalised access frequency is used as input variable for the clustering algorithm. However, there may be more factors that can influence the attractiveness than only the normalised access frequency. For example, in Experiment 1 Hypothesis 1, it is shown that more recently created files are more likely to be accessed than older files. This experiment only considers the aggregate function of access frequency instead of considering the reason why this access frequency is the way it is.

The reason to do so is the requirement of the EM algorithm to consider more than one attribute only when these prove to be independent of each other. This cannot be guaranteed and, therefore, only the aggregate function is chosen as input.

6.2.3 Experiment 3: Dynamic Likelihood

In Experiment 3, it is checked per hour if data sets are likely to be accessed jointly. Furthermore, after rules have been generated, they are checked whether they are statistically significant. All together, 134 significant association rules have been found on different levels of generalisation.

The fact that association rules are found indicates that data sets are not

Cluster	Mean	St. Dev.	Inst.	Perc. (%)
0	0.0383	0.0326	53	34
1	0.1729	0.0729	46	29
2	0.5387	0.1946	35	22
3	1.0301	0.0264	9	4
4	1.3155	0.5354	15	9

Table 6.4: Clusters of Attribute Survey

completely independent from each other. Hence, the hypothesis is confirmed: the likelihood of file B changes through the access of file A if B is associated with A (for a definition of association see Section 5.5, starting at page 34).

A table with all statistically significant association rules can be found in Appendix D. These rules can be used to form the basis for a variative value for (groups of) files to complement the basic value formed by the results of Experiment 1 and Experiment 2.

Discussion The rules as found so far prove to be statistically significant. However, the support of most of the found rules is relatively low (sometimes only 2 occurrences), although most of the rare rules have been filtered out. Although very rare rules might be useful in some cases, generally it will not heavily improve the hit rate of a replacement algorithm. Additionally, it does increase the overhead through running the algorithm.

Furthermore, although relations have been analysed on different levels of abstraction (file type, directory, survey and combinations thereof), the spatial component has not been added yet. Hence, these rules form the input for the next sub-hypothesis for further generalisation.

Further Generalisation, Sub-hypothesis: *The cause of the association between accesses of files is a small spatial distance.*

This sub-hypothesis is tested by means of creating matrices with the antecedents and consequences of all rules. The distance between the elements is compared to the distances in a matrix in which all surveys are represented. The results of this comparison are given in Appendix E.

The conclusion of this process is that the associated attributes (elements in the rules) are not associated due to a small distance. In some of the rules it is the case, but in most of the rules it is not. Therefore, a general conclusion that files are association due to a small distance between the surveys they both belong to can not be derived from this experiment. Therefore, the rules can only be used as they are right now. Further generalisation is not possible on spatial distance.

6.3 Data Mining Prefetching Algorithm

The results of the three experiments indicate that they can be used to formulate a prefetching algorithm which uses evidence from historical accesses to estimate probable future accesses. We call the algorithm the Data Mining Prefetching (DMP) algorithm

The forming of the algorithm is inspired by Markov chain modeling and conditional probability-based Bayesian networks [4] [7] [8] [9] [29] [19]. As already indicated in section 4.6.1 on page 24, these researches use the conditional

probability, which is the result of data mining techniques to model prediction of future activities. However, Markov chain modeling and Bayesian networks are very heavy methods. A huge number of possible states exist for this database, while many of them are useless and will probably not occur[2]. Markov chain modeling and Bayesian Networks are to capture all these states in a predictive algorithm. However, it is chosen not to use *all* knowledge in advance. That means that the knowledge is called upon when appropriate. As such, the algorithm does not require searching through all knowledge at all transactions.

The following subsection 6.3.1 describes the algorithm in words, whereas subsection 6.3.2 does so in Pseudo Code[3]. Some parts of the complete code that simulates both algorithms are given in Appendix F.

6.3.1 Algorithm in words

All files in the digital archive contain a basic value. This basic value is calculated by means of the results of Experiments 1 and 2. Experiment 1 determines whether a file is not attractive, whereas Experiment 2 distinguishes files on the *rate* of attractiveness.

Basic Value Experiment 1, Hypothesis 1 is used to create a 0 basic value for raw files when idented files are present within the same survey and the creation dates of the idented files are more recent than Jan 1st 2005.

Experiment 1, Hypothesis 2 is used to identify the most recent files (within three months distance from absolutely the most recent file). Each file within that group gets the number of Confidence divided by the size of the group. That means that if 5 files are in the group of most recent files and 20 files exist of that particular file type, the most recent group separately get a value of $\frac{Confidence}{5}$ and the other files get the value of $\frac{1-Confidence}{15}$.

From Experiment 1, hypothesis 2, the means μ of the clusters are used. Each attribute contributes for its part to the basic value of each file.

Together, the basic value is calculated by adding the contribution for recency of the file and the three cluster contributions and divide it by 4 (each contribution to weigh equally). When hypothesis 1 shows the need for diminishing the value, it is multiplied by 0 and otherwise by 1.

All files now have a basic value. This value is a weighted, general, static attractiveness of a file. A basic value of 1 indicates that it is likely that within the next period of the same length as the training period the file will be accessed once. However, as in the prefetching algorithm the values are compared, the number actually is a relative value instead of a real likelihood.

[2] only a small percentage of all files is accessed in 2006.

[3] Pseudo code is a brief description as in programming code but understandable through the use of natural language elements.

Variation Variation in the value of a certain file occurs when files are accessed that are present in one of the antecedents of the 134 association rules. If the antecedent is true (all predicates valid), the consequent files must increase in value. The value is calculated by multiplying the Confidence of the rule by a standard value.

Valid Files A file becomes 'valid' if it is accessed. Validation decreases in time. If a file is valid it can (maybe together with other files) put an antecedent value on true.

Total Value The total value per file is calculated by adding the basic value to the variative value. The highest value is the most probable to be accessed in the next transaction compared to all other files.

Parameters

Periods Each period contains 15 minutes. Hence, every 15 minutes it is checked which files have been accessed and which values must be updated.

When a file is accessed, the algorithm does not keep the file in the cache; it immediately releases the file after having delivered it. However, the algorithm does take into account a period of 1,5 hour as bridging period. This bridging period means that files remain valid for 1,5 hours.

Constraints The disk environment contains a total of 1 TeraByte. The algorithm provides a candidate cache to fill it up to 90%. However, the disk itself cannot extend 80% at the end of each period[4]. This also means that every file that has been archived is taken into account in this calculation. The same is counts for the limited amount of tape drives (6 tape drives with a speed of 30 MegaBytes/s).

Variation The maximum variative value is twice the maximum basic value. As such, files with a low basic value can still be candidates for the cache if they occur in a strong rule (high Confidence).

6.3.2 DMP in Pseudo Code

In this section, the algorithm is briefly described in Pseudo Code. The Pseudo code is given in Figure 6.3. In Appendix F part of the code is given. This code describes the same as is described in the previous section in words.

[4] 80% is a standard value within Shell to not decrease disk performance and keep available capacity for archive and access requests.

```
Do While transaction_list <> Empty

        Action 1:  Diminish validity of files accessed less then 1,5 hours ago
        Action 2:  Calculate performance

        If the transaction <> empty then
                Action 3:  Upgrade validity of all accessed files
                Action 4:  Search for relevant rules related to the accessed files

        Go through all relevant rules, if antecedent = valid Then
                Action 5:  Update values of files with attributes of the consequent
                           Delete the rule

                If the variation values have changed since last transaction then
                Action 6:  Make candidate caching (highest values)

                If candidates are not in cache
                        Do While constraints are not violated
                                Action 7:  Put in cache
                        Else keep the current cache
                End If
        End If

        Next transaction

End While
```

Figure 6.3: Pseudo Code DMP algorithm

6.4 Performance Comparison

Hit rate The hit rate of the DMP algorithm has proven to be 0.87%, the hit rate of the LRU simulation 25.58%. With respect to the hit rate, the LRU algorithm outperforms the proposed algorithm.

Overhead Compared to the proposed algorithm, the LRU algorithm is also much better with respect to overhead of the algorithm. The main overhead lies in searching through all files. The LRU algorithm only requires actions that are comparable with Action 2 and 7 of the DMP algorithm as described in section 6.3, whereas Action 1 and 3 need to search through *all* files. Action 5, 6 and 7 also search through all files, but these are not mandatory for all transactions. In certain transactions, the algorithm must search through more than a million files.

Overall Comparison The LRU algorithm outperforms the proposed algorithm, which is based on the results of the data mining process simulated on the test data. This is the case for both the overhead that the algorithm counts for and in terms of hit rate.

Discussion Based on the results from the previous section 6.4, several elements should be noted to discuss the outcomes.

Firstly, the outcomes indicate that the main assumption, the fact that the LRU algorithm is not optimal and can be improved, does not hold for the test data set.

The results also indicate that the training data set might be very different from the test data set. Therefore, the two sets (training and test) are now compared regarding the accesses per directory and survey. The graphs are given in Figures 6.4, 6.5 and 6.6.

The Figures show that the distribution of directories and surveys in 2007 is more extreme than in 2006. In both years, certain data sets are very popular. The distributions look comparable, but the distribution is more extreme (less data sets, larger differences) in 2007. For example, in the three months of 2007 the maximum is slightly above 5000 accesses, whereas during the 10 months in 2006 this maximum is slightly above 3000.

Especially when changes in attractive surveys and directories occur with the change from 2006 to 2007, the association rules completely loose their value. This might have been avoided if the 'Data Understanding' phase in Figure 3.1 on page 15 had been executed more thoroughly. The training and test data set could have been compared in advance.

Another element that can be concluded is that the users de-archive certain files several times during a short period. Hence, the archive behaves as intertwined with the normal working storage environment. This is not the goal of the current system and does not reflect the current set-up. The situations of before January 1st 2007 and thereafter differ with respect to the the ratio of uniquely accessed files to all accesses: 79% for 2006, 62% for 2007. So, during the testing period, more files were de-archived multiple times.

The value of association rules to propose a prefetching algorithm would increase heavily if the rules could be generalised better. The ratio of accessed files to the total amount of files is very low (around 10%) for the whole training set. That means that in year A, certain files may be interesting, whereas in another years other files are interesting. If the rules focus on instances of directories, surveys and file types, it may lead to inappropriate rules compared to more generalized rules. One can say that the learning mechanism shows overfitting, so the relations are too specific.

This also leads to the conclusion that association rules extracted from experimental data from a certain period do not guarantee that these rules also count in another period. The LRU algorithm in itself has the feature of focussing more on the *current* (attractive) files, which proves to be very powerful in the

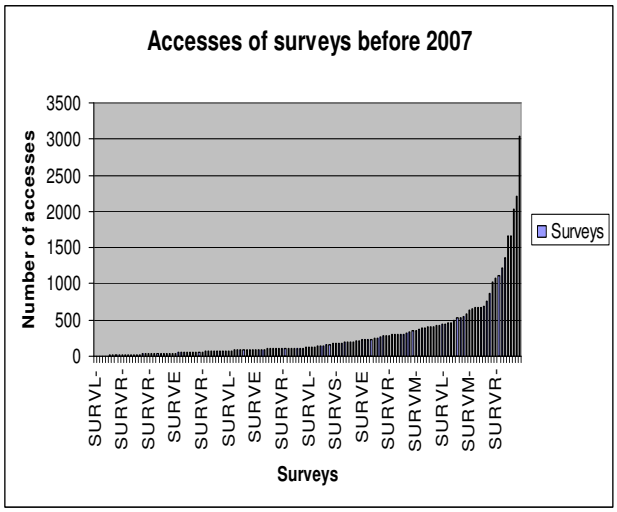

Figure 6.4: The accesses of files within surveys in 2006

test data set in the experiment. A caching algorithm may be improved if the rules extracted by data mining are adapted in time. This can be executed by establishing a data warehouse and execute the data mining process on a regular basis. This is also proposed by the authors in [1], [5], [9] and [17].

The results and this consideration also lead to the remark that the experimental period is either too long (one could focus only on short-term attractiveness) or too short to capture (all) relevant, valuable association rules.

With respect to the algorithm itself, some elements should be discussed as well. Whereas in the process of data mining, likelihoods are extracted, these likelihoods are not used as such. The likelihoods are converted into a value that stresses the relative value between files.

Another weakness is that the relations between these likelihoods (conditional probabilities) are not checked but might indicate that the weights for variative value calculation of files is not appropriate.

We intended to adapt the weights slightly for performance improvement, but the distance between the performance of the DMP and the LRU algorithm shows to be too large to put more efforts in improving it.

Although the performance with respect to the overhead is not emphasised heavily due to the big difference in hit rate performance, there are improvement opportunities. All files are currently considered without any indexing and smart

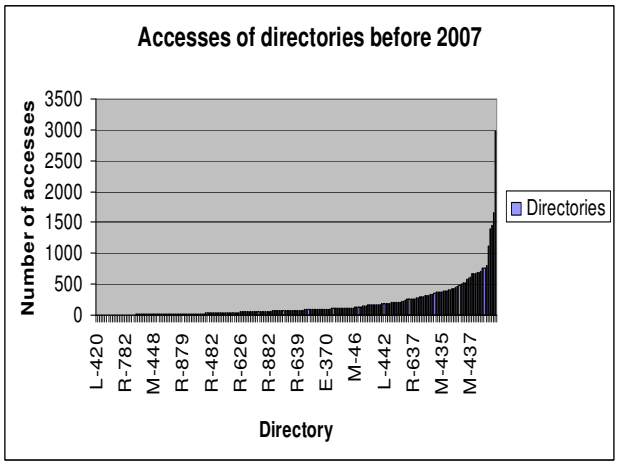

Figure 6.5: The accesses of files within directories in 2006

searching algorithms. That could shorten the searching time and hence, improve the overall performance of the system.

A final remark that is worthwhile to mention is that establishing the LRU algorithm and incorporating results from data mining are not mutually exclusive. It was already stated in the motivation in this research, Section 3.3 on page 10 that some researchers adapt and extend the LRU algorithm instead of *challenging* the algorithm. It might have been useful to incorporate that element in the proposed algorithm as well. However, a comparison does show more specifically the downside of completely replacing the LRU with the DMP algorithm and is, thus, also valuable.

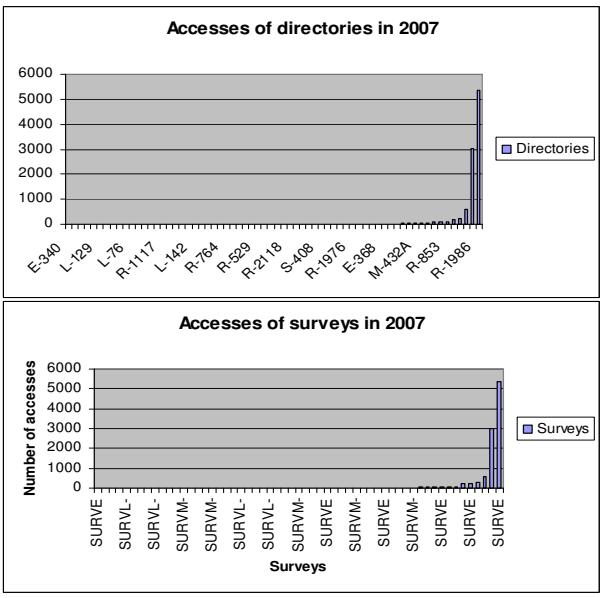

Figure 6.6: The accesses of files within directories and surveys in 2007 (test-set)

Chapter 7

Conclusions

7.1 Introduction

The goal of this chapter is to provide the answers to the research questions as posed in Section 3.4 on page 13. Furthermore, the aim is to discuss the outcomes, review the choices made during the research and to provide areas for further research.

In Section 7.2 the research questions are answered. Subsequently, in Section 7.3 the research is discussed and reviewed. Section 7.4 provides elements for further research following on this research. The chapter's final section 7.5 shows what broader implications of this research are.

7.2 Research Questions

7.2.1 Question 1

The main selection criteria to assess which data mining techniques to use are easy understandable knowledge representation, little algorithm overhead and an unsupervised manner of learning from historical access data.

For general attractiveness of files, it is chosen to use statistical measures to assess two explicit hypotheses. These two hypotheses state that the likelihood of certain files is decreased by the presence of others in the archive. The first hypotheses assesses the relative attractiveness of idented and raw files. The second assesses the relative attractiveness with respect to recency of the creation date of files.

Additionally, clustering is used to analyse the normalised frequency of the attributes data type, directory and survey. Specifically, the Expectation Maximisation clustering algorithm is used as it does not require determination of the number of clusters in advance.

To analyse joint accesses, Association Rule Mining is used. The Apriori algorithm with very low metrics thresholds is used for rule generation. Subse-

quently, rules are checked on their statistical significance with a confidence level of 99,85%.

7.2.2 Question 2

Two explicit hypotheses are both confirmed. From these hypotheses, it can be concluded that idented files with a creation date of less than two years ago are more likely to accessed compared to raw files from the same survey. Furthermore, there exists a preference to access the most recent data set when more data sets are available from the same survey from the same data type.

The Expectation Maximisation algorithm has found more than one cluster for each attribute, indicating that the general attractiveness is determined by the three tested attributes.

The Apriori algorithm has found many association rules on different levels of the data taxonomy. This taxonomy is extracted from the research database, which is constructed by means of interviews with domain experts. After having found the rules and checking them on statistical significance, 134 association rules are left. These rules indicate that it is not coincidence that certain files are accessed jointly. These rules can predict a sequence during time $(t + \tau)$ based on accesses at time t, τ being small.

7.2.3 Question 3

A prefetching algorithm Data Mining Prefetching (DMP) is formulated that uses both the general attractiveness and the increased attractiveness (access likelihood) due to association. The former attractiveness results in a basic value, whereas the latter results in a variative value. Jointly, they form a total value, which determines which files should be present on primary storage.

The DMP algorithm updates primary storage every 15 minutes. It also takes into account the constraints of the system: disk space maximum and tape drive capacity. This causes the amount of prefetched files to be dependent on traffic as is proposed as well in [2] and [11].

7.2.4 Question 4

The DMP algorithm is compared to the Least Recently Used algorithm, which is currently used for the digital archive. Performance is compared on two metrics: the hit rate and the computing overhead.

The LRU algorithm outperforms the DMP algorithm on both metrics. With respect to the hit rate, the LRU algorithm performs around 30 times better than the new algorithm (0,87% hit rate compared to 25,58%). A quantitative analysis regarding the computing overhead is not executed. It is argued though, that the major overhead is caused by searching through the files. The LRU algorithm does not have to do so, whereas the new algorithm has to do so several times

per period of 15 minutes. For certain transactions the DMP algorithm must search through more than a million files, which is not required for the LRU algorithm.

7.2.5 Main Question

In this research the currently used replacement algorithm is compared to a prefetching algorithm, based on the results of a data mining process. The process shows the potential of using the results as it comes up with differing general values and statistically significant rules of data set associations.

However, the LRU algorithm heavily outperforms the proposed algorithm. This is (partly) caused by the (sudden) change of the test data set compared to the training data set. Furthermore, the rules extracted by using data techniques mining should be either updated regularly or generalised more to be of more value.

The results indicate that an algorithm based on the results of data mining should not replace the LRU algorithm to improve performance.

7.3 Discussion

The outcome of this research is a prefetching algorithm, based on the results of data mining on historical access data and a performance comparison of this algorithm with the currently used algorithm. From the analysis of this comparison, we see that the practical added value of the new algorithm is not present. Therefore, we see a negative answer to the posed Main Research Question. However, former research shows better results with comparable methodologies. This section, therefore, reviews some of the choices made in this research.

7.3.1 Methodology

In this research, it is chosen to use statistical metrics, clustering techniques and association rules. The choice is funded by literature and former research shows the use of similar techniques. However, the combination of such techniques is not seen frequently. In the newly proposed algorithm, the outcome of these results are combined in a fairly arbitrary way. This can lead to a situation in which the outcomes strengthen each other, leading potentially to a biased weighting. There could have been executed more research about the combination of values and the way to do so in an eventual algorithm. Furthermore, iteratively testing the weights between using the separate results may have lead to a better result.

7.3.2 Execution

It is argued in section 6.4 that when one is unable to generalise association rules as much as possible, the value of association rules within a digital archive is not

so high, unless the time of research is at least as long as one business cycle. A business cycle is then defined as a period in which (almost) all data sets are accessed once. Another way would be to shorten the association rule extraction periods. As such, one can use the data sets that are attractive for that particular moment. However, the best way would be to generalise the rules, so that they will count for a large percentage of the whole archive for a long period.

Furthermore, association rules are assessed on statistical significance with a confidence level of 99,85%. This level has not been adapted, causing the possibility of the level to be too strict. There might have been found more association rules when this level had been adapted.

It is tried to generalise the rules as much as possible. Different levels of abstraction are used during the rule mining process. Subsequently, the association rules are assessed on spatial distance. Unfortunately, this relation was not found. No other relations have been checked, as this was the most logical one to do research on. However, one could argue that it would have been valuable to go back to domain experts to discuss which other generalisation opportunities there might be. These extra efforts could have improved the prefetching algorithm.

7.3.3 Conceptually

In this research, the LRU algorithm is challenged by proposing a completely new algorithm. It is based on a thought process and situational factors of the system within Shell. However, whereas in this research the analyses are executed as separate algorithms, it is worthwhile to incorporate data mining as an *addition* to the LRU algorithm. That could have given a better answer to the main research question what data mining results can do to *improve* the algorithm of a digital archive. This is also proposed in [3]

The reason that this has not been executed is the fact that a more detailed research on the value of the LRU algorithm is required if one requires a combinatory algorithm. Otherwise, it is impossible to argument which of the two algorithms should be dominant and what the difference in weights between the two should be.

A final remark in this section is made to underline the assumptions as made in the Introduction. It is assumed through executing interviews with users that they communicate well and use de-archived data together. However, this assumption may not be completely true. The results of uniquely de-archived files indicate that this might be even untrue.

7.4 Further Research

Further research could be useful in the following areas:

Changes in usage It is useful to extract cluster values and association rules during differing time periods. For example, one could look per month, per quarter of a year, per three years etc. As such, one could get a feeling of changes in usage and one could use different (sorts of) rules in different periods.

LRU algorithm The results indicate that the LRU algorithm has a good performance on the test data set. It is valuable to test whether this is always the case.

Furthermore, if one would be able to combine the LRU algorithm with data mining results, it is valuable to know what time period commonly lies between accesses. Are files always accessed a day after the previous access, if they are accessed? Or does it take longer? Is it always the same? If these questions could be answered, one can extend and improve the LRU algorithm to work even more effectively. Furthermore, weights could be identified to which extent a time-based solution like the LRU should be used compared to a data mining based algorithm.

Geophysical Content This research is executed for a thesis Artificial Intelligence. Therefore, several assumptions are made to construct the research database. However, due to a lack of precise understanding, some relations may be overlooked. Therefore, detailed discussions about the assumptions made with domain experts could improve the research database and as such, the data mining process due to a better data taxonomy.

7.5 Broader Implications

Archiving as is executed within Shell is a form of active, structured archiving. The actual archive is not very structured (see Appendix A for explanation), but the data management is. Digital archiving is a business area that rises in a fast pace, mainly driven by regulatory forces and the explosion of the amount of digital data. Currently, research regarding digital archiving focusses on reliable, efficient and cheap, mostly hardware-driven solutions. However, the field of (web-)caching shows the potential for prefetching algorithms based on data mining results. If one can improve the software part, the problem of digital archiving is attacked from another side and will therefore improve.

For some large websites, the separate parts of websites are also store hierarchically. This new method of prefetching might introduce new opportunities for such websites. These large websites merely contain large multi-media elements.

For Shell as supervisor of this research, it is valuable to get an overview of digital archiving, the problems that arise with it and the data management efforts underlying it. It emphasises the need for good connections between data sets, which seems not too easy to achieve. Furthermore, it sketches the

problem of different databases which are not aligned with each other, nor with the business process that they serve. A data warehouse, both for common use and data mining efforts would benefit the situation.

Furthermore, Shell can learn from this research that the current system set-up does not reflect the business process. The set-up suspects there being an 'active archiving', but it seems that users use the archive intertwined with their active storage environment. Shell could use this information to either adapt the processes slightly and, therewith, protect the system from uselessly using too much system capacity. It could also adapt the system and integrate the system better with the working storage environments of the users.

Bibliography

[1] Amo, S. de, Furtado, D.A., 2007. First-order temporal pattern mining with regular expression constraints. *Data & Knowledge Engineering,* in press

[2] Balamash, A., Krunz, M., Nain, P., 2007. Performance analysis of a client-side caching/prefetching system for Web traffic. *Computer Networks,* in press

[3] Bonchi, F., Gianotti, F., Gozzi, C., Manco, G., Nanni, M., Pedreschi, D., Renso, C., Ruggieri, S., 2001. Web log data warehousing and mining for intelligent web caching. *Data & Knowledge Engineering,* 39: 165-189

[4] Borges, J., Levene, M., 2004. An average linear time algorithm for web usage mining. *International Journal of Information Technology & Decision Making,* 3(2): 307-319

[5] Damle, C., Yalcin, A., 2007. Flood prediction using Time Series Data Mining. *Journal of Hydrology,* 333: 305-316

[6] Elmasri, R., Navathe, S.B., 2003 . *Fundamentals of Database Management Systems,* New Jersey: Addison Wesley

[7] Gámez, J.A., Moral, S., Salmerón, 2004. *Advances in Bayesian Networks*. Berlin: Springer

[8] Giudici, P., Passerone, G., 2002. Data mining of association structures to model consumer behaviour. *Computational Statistics & Data Analysis,* 38: 533-541

[9] Ha, S.H., Bae, S.M., Park, S.C., 2002. Customer's time variant purchase behavior and corresponding marketing strategies: an online retailer's case. *Computers & Industrial Engineering,* 43: 801-820

[10] Hong, T-P., Lin, K-Y., Wang, S-L., 2003. Fuzzy data mining for interesting generalized association rules. *Fuzzy sets and Systems,* 138: 255-269

[11] Huang, Y.-F., Hsu, J.-M., 2007. Mining web-logs to improve hit ratios of prefetching and caching. *Knoledge-Based Systems,* in press

[12] Keller, G., Warrack, B., 2003. *Statistics for Management and Economics*. California: Thomson Learning

[13] Larose, D.T., 2005. *Discovering Knowledge in Data, An Introduction to Data Mining.* New Jersey: John Wiley & Sons

[14] Larose, D.T., 2006. *Data Mining, Methods and Models.* New Jersey: John Wiley & Sons

[15] Mookerjay, V.S., 2002. Policies for data archival in hierarchical storage management. *Computing, Artificial Intelligence and Information Technology,* 138: 413-435

[16] Reddy, M., Fletcher, G.P., 1998. Exp1: a comparison between a simple adaptive caching agent using document life history and existing caching techniques. *Computer Networks and ISDN Systems,* 30:2149-2153

[17] Roszypal, A., Kubat, M., 2005. Association mining in time-varying domains. *Intelligent Data Analysis*, 9: 273-288

[18] Russel, S.J., Norvig, P., 1995. *Artificial Intelligence, a modern approach.* New Jersey: Prentice Hall

[19] Santana, A.L., Francês, C.R., Rocha, C.A., Carvalho, S.V., Vijaykumar, N.L., Rego, L.P., Costa, J.C., 2007. Strategies for improving the modeling and interpretability of Bayesian networks. *Data & Knowledge Engineering,* in press

[20] Shekhar, S., Chawla, S., 2003. *Spatial Databases, A Tour*. New Jersey: Prentice Hall

[21] Shenoy, P.D. Srinivasa, K.G., Venugopal, K.R., Patnaik, L.M., 2005. Dynamic association rule mining using genetic algorithms. *Intelligent Data Analysis,* 9: 439-453

[22] Shotong, W., Chung, K.F.L., Hongbin, S., 2005. Fuzzy taxonomy, quantitative database and mining generalized association rules. *Intelligent Data Analysis,* 9: 207-217

[23] Silberschatz, A., Galvin, P.B., Gagne, G. *Operating System Concepts with Java.* New Jersey: John Wiley & Sons

[24] Srikant, R., Agrawal, R., 1997. Mining generalized association rules. *Future Generation Computer System,*: 13: 161-180

[25] Tseng, M-C., Lin, W-Y., 2007. Efficient mining of generalized association rules with non-uniform minimum support. *Data & Knowledge Engineering,* in press

[26] University of Waikato. *Weka,* http://www.cs.waikato.ac.nz/ml/weka/

[27] Vakali, A., 2000. Data block prefetching and caching in a hierarchical storage model. *Information Sciences,* 128: 19-41.

[28] Witten, I.H., Frank, E., 2005. *Data Mining, Practical Machine Learning Tools and Techniques.* San Francisco: Morgan Kaufman Publishers

[29] Ye, N., 2003. *The Handbook of Data Mining.* New Jersey: Lawrence Erlbaum Associates

Appendix A

Databases

One of the motivations of this research is a preceding research about the feasibility study of a next-generation digital archive system. This system within Shell is called the HSM, referring to the concept of Hierarchical Storage Management.

In the next sections, the databases that are used within this research are described briefly. Furthermore, the relations between the separate databases/systems are described.

A.1 HSM

The HSM system is the archive of seismic and related data. The structure of the HSM is fairly straightforward. This structure can be found at several other organisations as digital archive. Among them is SARA, the scientific network operator and tier-1 archiving and processing center for the CERN particle accelerator project.

The HSM contains two storage layers, a disk and a tape environment and software to migrate data between disk and tape on the basis of the LRU algorithm. The disk environment behaves as a regular Unix File System of which the most important partition is organised in a flat manner. That means that it contains directories (all on the same level) with files therein. The directory names are codes, which refer to another system called eProject. The codes sometimes refer to processing phases. However, the files within the directories are almost always variative.

The HSM records everything that happens regarding accesses and archives. It collects these data in the form of logs (reports).

A.2 eProject

eProject is an on-line (Shell) in-house developed web-tool that runs on the intranet within Shell. It is designed as a project management tool. It contains elements of time lines, stage gates and an archiving component as well.

The projects are organised on the basis eProject codes. Every time a project starts it receives a code by which it is referred to the type of project. However, the nature of the project sometimes alters, so the elements of the code do not always correctly reflect the project contents.

The projects not only contain this code but also have names. These names mostly refer to place on the earth if it is land data and to names of blocks if it is marine data. The north sea is divided in blocks which contain names and the seismic data set are named after these blocks. These names, the eProject codes are sometimes in an Oracle based database in which coordinate reference data is stored. This is (incorrectly) called ArcGis in this thesis and is explained in the next section.

A.3 ArcGis

AcrGis is a name of a product that is licensed by Shell and developed by a data management application company called ESRI. Within Shell, a global team exists called the Geomatics Team. This team is concerned with everything that has to do with data management with (visual) maps.

ArcGis is a GIS (Geographical Information System) based piece of software. If a(n Oracle) database is enabled with spatial objects, ArcGis is capable of visualising the spatial objects on a map.

In this research, the visualisation part is not necessarily interesting, but the coordinates are. The database which lies under ArcGis has stored almost all seismic acquisition data sets.

In this database, the eProjects are sometimes (read: definitely not always) present and sometimes a name shows analogies with the names in the eProject application. However, this connection is not made automatically and is not checked by neither of the responsible teams.

A.4 EGIS

EGIS is an abbreviation for Exploratorion Geographic Information System. The name seems to refer to the same functionality as ArcGis in section A.3. However, this is definitely not the case.

EGIS is an Oracle based database in which meta data is collected of everything that is in the HSM system. That is, the EGIS instance that is interesting for this research. Worldwide, several instances are present which all more or less do the same. However, a system like the HSM is only present in Europe.

EGIS describes all files that are in the directories of the HSM. For this research, this means the data type and the place where one can find the file on the HSM. Hence, the directoryname (eProject) is connected to the files. Sometimes a name is also connected to the files, which can be comparable with the names in the ArcGis system and the eProject system.

A.5 Data Collection

Through manually connecting these four sources of data (management), the data is gathered to fill up the database for this research. As already indicated in some of the previous sections, formal connections between the sources do not exist, so everything had to be done manually. Furthermore, some of the connections are just impossible to make by the researcher. All together, meta data of around 20% of all data from the HSM is gathered to fill up the database.

A.6 Research Database

The database for this research is designed on a Linux version of PostgreSQL. This is an open-source database application server. Additionally, packages and libraries are installed. The main additional package is the PostGIS package, which enables the database to work with spatial objects. Several objects are defined, of which the polygon is the most important. In this way, coordinates are made possible to compare by means of spatial queries. Examples of spatial queries are overlapping, touching, union, within and distance. More information can be found on http://www.postgresql.org and http://www.postgis.refractions.net.

Appendix B

Data Types

Table B shows the meaning of the codes which refer to data types. The fact that more data types are present here than in section 6.2.1 on page 37 means that not all data types that are present in the archive are accessed during the training period.

Code	Explanantion
QCREP	MASS TRANSCRIPTION QC REPORT
RMOPIC	EDITED RMO PICKS
SAF	SEISMIC ATTRIBUTE FILES
SATIMA	SATELITE IMAGE
SEICOL	SEISMIC COLOUR SECTION
SPS	SPS DATA
SPSPRO	SPS PROCESSED DATA
SPSRAW	SPS RAW DATA
SRLPD	SHORT REFRACTION DATA (PAPER)
STK	STACK
STKDMO	DMO STACK
STKDZO	DZO STACK
STKFLT	FILTERED STACK
STKLPS	STACKLAMPS
STKMIG	STACK PLUS MIGRATION
STKMIN	MINIMUM PHASE STACK
STKNMO	STACK NMO ZEROPHASE
STKUNF	UNFILTERED STACK
STKUNP	STACK WITHOUT POST STACK PROCESSING
STKZER	ZERO PHASE STACK
STT	STATICS

Continued Code	Explanantion
STTFLD	FIELD STATICS
STTPRO	PROCESSED STATICS
STTRES	STATICS RESIDUAL
TES	TEST
TESPRO	PROCESSED TEST
TESRAW	RAW TEST
TOACPA	BACK-UP TOPOGRAPHICAL & ACQUISITION PAPERDATA
TOPO	TOPOGRAPHICAL DATA
TRAREP	TRANSCRIPTION REPORT
TSTQC	QC TEST DATA
VARREC	VAR RECORDS (PAPER)
VEL	VELOCITY
VELANA	VELOCITY ANALYSIS
VELMIG	MIGRATION VELOCITIES
VELMOD	MODELLING VELOCITIES
VELMOV	MOVIE FILE
VELPRM	PRE STACK TIME MIGRATION VELOCITIES
VELPSI	PSI VELOCITY
VELSTK	STACKING VELOCITIES
WEL	WELL LOG DATA
WELDIP	WELL DIPMETER MSD
WELPD	WELL PAPERDATA
WELPRO	PROCESSED WELL
WELRAW	RAW PRODUCTION WELL
WELVEL	WELL VELOCITIES

Table B.1: Codes and explanation data types

Appendix C

Cluster Members

In this appendix, the tables with cluster members are given for all three attributes for which clusters have been defined. These results are derived from running the Expectation Maximisation algorithm on three different attributes: data type, directory and survey.

C.1 Data Type

cluster0	cluster1
ACQPD	GATPRO
ACQREP	MAP
ASCCOV	MIG
ASCSTT	MIGDEP
COV	MIGDMO
FLDIDT	MIGDTM
FLDRAW	MIGGAT
GAT	MIGNMO
GATPSI	MISCEL
MAPCOV	POSSDM
MAPLOC	PRESDM
MIGPSI	PRETIM
MIGZER	PROREP
NAVPRO	STK
PROSUP	STKDMO
STKLPS	STKFLT
STT	STKNMO
TRAREP	STKUNF
TSTQC	STKZER

Continued cluster0	cluster1
	STTFLD
	VEL
	VELMIG
	VELMOD
	VELPRM
	VELSTK

Table C.1: Clusters of Attribute Data Type

C.2 Directory

cluster0	cluster1	cluster2	cluster3	cluster4
E-343	E-344	E-378	R-1976	E-340
E-348	E-346	L-2107	R-605	E-358
E-349	E-350	R-1012	R-607	E-373
E-356	E-351	R-1112	R-629	E-379
E-360	E-363	R-1280	R-637	E-384
E-362	E-366	R-1592	R-687	E-387
E-381	E-368	R-2015	R-89	E-623
E-389	E-369	R-497		L-104
E-577	E-370	R-498		L-105
E-578	E-375	R-505		L-107
L-106	E-376	R-532		L-108
L-420	E-383	R-536		L-109
L-427	E-386	R-545		L-118
L-455	E-696	R-548		L-129
L-464	L-463	R-554		L-134
L-470	R-1070	R-561		L-137
L-480	R-1521	R-590		L-142
L-481	R-1607	R-603		L-301
L-483	R-1986	R-609		L-302

Continued

cluster0	cluster1	cluster2	cluster3	cluster4
L-485	R-2011	R-627		L-327
L-495	R-2020	R-639		L-330
L-504	R-2118	R-653		L-436
L-508	R-511	R-655		L-44
L-523	R-560	R-661		L-442
L-550	R-563	R-703		L-446
L-602	R-843	R-818		L-468
L-645	R-883	R-826		L-51
L-652A		R-858		L-52
L-724		R-861		L-76
M-1021		R-868		L-83
M-108		R-879		L-84
M-432B		R-881		L-99
M-433		R-900		M-309
M-438		R-940		M-329
M-441				M-432A
M-443				M-435
M-448				M-437
M-452				M-444
M-472				M-451
M-473				M-46
M-474				M-461
M-487				M-471
M-488				M-502
M-494				M-527
M-526				M-591
M-528				M-648
M-55				R-1003

Continued

cluster0	cluster1	cluster2	cluster3	cluster4
M-581				R-1008
M-625				R-1011
M-649				R-1024
M-91				R-1034
M-98				R-1045
R-1004				R-1047
R-1005				R-1059
R-1025				R-1061
R-1038				R-1117
R-1041				R-1279
R-1072				R-1454
R-1273				R-1540
R-1806				R-1723
R-1813				R-1776
R-1983				R-1782
R-516				R-1822
R-600				R-482
R-697				R-501
R-721				R-529
R-754				R-540
R-775				R-541
R-776				R-547
R-782				R-568
R-805				R-575
R-810				R-576
R-813				R-582
R-829				R-593
R-830				R-604
R-831				R-626
S-447				R-636

Continued cluster0	cluster1	cluster2	cluster3	cluster4
S-465				R-664
S-486				R-668
S-489				R-670
S-491				R-672
S-492				R-681
S-493				R-759
S-496				R-764
S-517				R-784
				R-789
				R-811
				R-853
				R-856
				R-870
				R-882
				R-888
				R-891
				R-897
				R-899
				R-939
				R-945
				S-19
				S-316
				S-408

Table C.2: Clusters of Attribute Directory

C.3 Survey

cluster0	cluster1	cluster2	cluster3	cluster4
SURVEL-373	SURVEM-349	SURVEL-368	SURVEM-348	SURVEL-696
SURVEM-350	SURVEM-358	SURVEL-369	SURVEM-360	SURVEM-340
SURVEM-387	SURVEM-384	SURVEL-370	SURVEM-362	SURVEM-343
SURVL-107	SURVEM-578	SURVEL-386	SURVEM-375	SURVEM-351
SURVL-118	SURVEM-623	SURVEM-344	SURVEM-376	SURVEM-356
SURVL-2107	SURVL-142	SURVEM-346	SURVEM-378	SURVL-420
SURVL-301	SURVL-327	SURVEM-363	SURVEM-381	SURVL-84
SURVL-468	SURVL-330	SURVEM-366	SURVEM-383	SURVM-494
SURVL-508	SURVL-442	SURVEM-379	SURVEM-577	SURVR-568
SURVL-51	SURVL-455	SURVEM-389	SURVL-104	
SURVL-52	SURVL-463	SURVL-106	SURVL-105	
SURVL-523	SURVL-464	SURVM-329	SURVL-108	
SURVL-550	SURVL-470	SURVM-471	SURVL-109	
SURVL-652A	SURVL-480	SURVM-528	SURVL-129	
SURVL-76	SURVL-481	SURVM-581	SURVL-134	
SURVL-83	SURVL-483		SURVL-137	
SURVM-1021	SURVL-485		SURVL-302	
SURVM-432B	SURVL-495		SURVL-427	
SURVM-438	SURVL-645		SURVL-436	
SURVM-451	SURVM-108		SURVL-44	
SURVM-452	SURVM-432A		SURVL-446	
SURVM-46	SURVM-441		SURVL-469	
SURVM-461	SURVM-443		SURVL-504	
SURVM-473	SURVM-444		SURVL-602	
SURVM-474	SURVM-472		SURVL-724	
SURVM-487	SURVM-591		SURVL-99	
SURVM-488	SURVM-649		SURVM-309	
SURVM-502	SURVM-91		SURVM-433	

Continued cluster0	cluster1	cluster2	cluster3	cluster4
SURVM-526	SURVM-98		SURVM-435	
SURVM-527	SURVR-1280		SURVM-437	
SURVM-55	SURVR-1592		SURVM-448	
SURVM-625	SURVR-516		SURVM-648	
SURVR-1012	SURVR-540		SURVR-1011	
SURVR-1045	SURVR-603		SURVR-881	
SURVR-1521	SURVR-604		SURVR-899	
SURVR-482	SURVR-672			
SURVR-541	SURVR-789			
SURVR-575	SURVR-870			
SURVR-629	SURVR-882			
SURVR-721	SURVR-900			
SURVR-784	SURVS-19			
SURVR-853	SURVS-408			
SURVR-856	SURVS-453			
SURVR-883	SURVS-489			
SURVR-897	SURVS-491			
SURVR-939	SURVS-492			
SURVS-316				
SURVS-447				
SURVS-465				
SURVS-486				
SURVS-493				
SURVS-496				
SURVS-517				

Table C.3: Clusters of Attribute Survey

Appendix D

Association Rules

In this appendix, the association rules are given. These rules are the result of running the Apriori algorithm and subsequently, assessing the statistical significance with a confidence level of 99,85%.

	Significant Rules with Confidence $\geq 50\%$
1	SURVM-461.PRESDM → SURVM-108.PRESDM
2	SURVEM-578.GATPRO → SURVM-46.GATPRO
3	SURVM-46.GATPRO → SURVEM-578.GATPRO
4	SURVM-108.MIGGAT Λ SURVM-329.FLDIDT → SURVM-461.MIGGAT
5	SURVM-329.FLDIDT Λ SURVM-461.MIGGAT → SURVM-108.MIGGAT
6	SURVL-83.FLDIDT → SURVL-83.GATPRO
7	SURVM-432A.GATPRO → SURVM-91.GATPRO Λ SURVR-1592.MIGGAT
8	SURVM-432A.GATPRO Λ SURVM-91.GATPRO → SURVR-1592.MIGGAT
9	SURVM-432A.GATPRO Λ SURVR-1592.MIGGAT → SURVM-91.GATPRO
10	SURVM-91. Λ SURVR-1592. → SURVM-432A.
11	SURVM-91.GATPRO → SURVM-432A.GATPRO Λ SURVR-1592.MIGGAT
12	SURVM-91.GATPRO Λ SURVR-1592.MIGGAT → SURVM-432A.GATPRO
13	E-350. Λ E-375. → E-623.
14	E-350.FLDIDT Λ E-375.FLDRAW → E-623.FLDIDT
15	E-375. Λ E-623. → E-350.
16	E-375.FLDRAW → E-350.FLDIDT Λ E-623.FLDIDT
17	E-375.FLDRAW Λ E-623.FLDIDT → E-350.FLDIDT
18	R-511.FLDIDT → R-511.GATPRO

	Continued 1 Significant Rules with Confidence $\geq 50\%$
19	SURVEM-350. Λ SURVEM-375. → SURVEM-623.
20	SURVEM-350.FLDIDT Λ SURVEM-375.FLDRAW → SURVEM-623.FLDIDT
21	SURVEM-375. → SURVEM-350. Λ SURVEM-623.
22	SURVEM-375. Λ SURVEM-623. → SURVEM-350.
23	SURVEM-375.FLDRAW → SURVEM-350.FLDIDT Λ SURVEM-623.FLDIDT
24	SURVEM-375.FLDRAW Λ SURVEM-623.FLDIDT → SURVEM-350.FLDIDT
25	M-432B.MIG → M-432B.FLDRAW
26	M-527.GATPRO Λ R-1592.STKNMO → M-648.FLDIDT
27	M-648.FLDIDT Λ R-1592.STKNMO → M-527.GATPRO
28	R-1592. Λ R-672. → R-670.
29	R-1592.MIGGAT Λ R-670.GATPRO → R-672.FLDIDT
30	R-1592.MIGGAT Λ R-672.FLDIDT → R-670.GATPRO
31	R-511. Λ R-661. → S-19.
32	R-511.FLDIDT → R-511.GATPRO Λ S-19.FLDRAW
33	R-511.FLDIDT Λ R-511.GATPRO → S-19.FLDRAW
34	R-511.FLDIDT Λ S-19.FLDRAW → R-511.GATPRO
35	R-511.GATPRO Λ S-19.FLDRAW → R-511.FLDIDT
36	R-670. Λ R-672. → R-1592.
37	R-670.GATPRO Λ R-672.FLDIDT → R-1592.MIGGAT
38	SURVL-83.FLDIDT Λ SURVL-83.GATPRO → SURVS-19.FLDRAW
39	SURVL-83.FLDIDT Λ SURVS-19.FLDRAW → SURVL-83.GATPRO
40	SURVM-527.GATPRO Λ SURVR-1592.STKNMO → SURVM-648.FLDIDT
41	SURVM-648.FLDIDT Λ SURVR-1592.STKNMO → SURVM-527.GATPRO
42	SURVM-91.GATPRO Λ SURVR-672.FLDIDT → SURVM-432A.GATPRO
43	SURVM-91.GATPRO Λ SURVR-672.FLDIDT → SURVR-1592.MIGGAT
44	SURVR-1592.MIGGAT Λ SURVR-672.FLDIDT → SURVM-432A.GATPRO
45	SURVR-1592.MIGGAT Λ SURVR-672.FLDIDT → SURVM-91.GATPRO
46	SURVR-672.FLDIDT → SURVM-432A.GATPRO Λ SURVR-1592.MIGGAT
47	SURVR-672.FLDIDT → SURVM-91.GATPRO Λ SURVR-1592.MIGGAT
48	E-578.ACQPD → E-578.FLDRAW
49	L-129.MISCEL → L-129.FLDRAW
50	M-472.MISCEL → M-472.FLDRAW

	Continued 2
	Significant Rules with Confidence $\geq$ 50%
51	M-472.MISCEL → M-473.FLDIDT
52	M-472.MISCEL Λ M-474.FLDIDT → M-473.FLDIDT
53	M-473.FLDIDT Λ M-474.FLDIDT → M-487.FLDIDT Λ M-494.GATPRO
54	M-473.FLDIDT Λ M-474.FLDIDT Λ M-494.GATPRO → M-487.FLDIDT
55	M-473.FLDIDT Λ M-487.FLDIDT Λ M-494.GATPRO → M-474.FLDIDT
56	M-473.FLDIDT Λ M-494.GATPRO → M-474.FLDIDT Λ M-487.FLDIDT
57	M-474.FLDIDT Λ M-487.FLDIDT → M-473.FLDIDT Λ M-494.GATPRO
58	M-474.FLDIDT Λ M-487.FLDIDT Λ M-494.GATPRO → M-473.FLDIDT
59	M-474.FLDIDT Λ M-494.GATPRO → M-473.FLDIDT Λ M-487.FLDIDT
60	M-487.FLDIDT Λ M-494.GATPRO → M-473.FLDIDT Λ M-474.FLDIDT
61	R-1003.FLDIDT Λ R-1592.STKNMO → R-891.FLDIDT
62	R-1003.FLDIDT Λ R-891.FLDIDT → R-1592.PRETIM
63	R-1003.FLDIDT Λ R-891.FLDIDT → R-1592.STKNMO
64	R-1592.STKNMO Λ R-891.FLDIDT → R-1003.FLDIDT
65	R-1813.PRESDM → R-1976.PRESDM
66	R-511.FLDRAW → R-511.GATPRO
67	SURVL-129.MISCEL → SURVL-129.FLDRAW
68	E-381.FLDRAW → E-381.VELSTK
69	E-381.VELSTK → E-381.FLDRAW
70	E-578.ACQPD → E-577.FLDIDT Λ E-578.FLDRAW
71	E-578.ACQPD → E-577.FLDIDT Λ E-578.FLDRAW Λ E-623.FLDIDT
72	E-578.ACQPD → E-577.FLDIDT Λ E-623.FLDIDT
73	E-578.ACQPD Λ E-578.FLDRAW → E-577.FLDIDT
74	E-578.ACQPD Λ E-578.FLDRAW → E-577.FLDIDT Λ E-623.FLDIDT
75	E-578.ACQPD Λ E-578.FLDRAW → E-623.FLDIDT
76	E-578.ACQPD Λ E-623.FLDIDT → E-577.FLDIDT
77	E-578.ACQPD Λ E-623.FLDIDT → E-577.FLDIDT Λ E-578.FLDRAW
78	E-578.ACQPD Λ E-623.FLDIDT → E-578.FLDRAW
79	L-504.FLDIDT → L-504.MIG Λ L-504.STK
80	L-504.FLDIDT Λ L-504.MIG → L-504.STK
81	L-504.FLDIDT Λ L-504.STK → L-504.MIG
82	L-504.FLDIDT Λ R-1986.FLDIDT → R-826.PRESDM
83	L-504.FLDIDT Λ R-826.PRESDM → R-1986.FLDIDT
84	L-504.MIG → L-504.FLDIDT
85	L-504.MIG → L-504.FLDIDT Λ L-504.STK
86	L-504.MIG Λ L-504.STK → L-504.FLDIDT
87	L-504.STK → L-504.MIG
88	L-504.STK Λ R-1986.FLDIDT → R-826.PRESDM

	Continued 3 Significant Rules with Confidence $\geq 50\%$
89	L-504.STK Λ R-826.PRESDM → R-1986.FLDIDT
90	R-1011.VELMIG → R-1011.VELSTK
91	R-1011.VELSTK → R-1011.VELMIG
92	R-1034.MIGNMO → R-1045.MIGNMO
93	R-1592.VELSTK → R-891.FLDIDT
94	R-482.FLDIDT Λ R-568.FLDIDT → R-540.GATPRO
95	R-482.FLDIDT Λ R-568.FLDIDT → R-575.GATPRO
96	R-511.FLDRAW → R-511.FLDIDT
97	R-540.GATPRO Λ R-568.FLDIDT → R-575.GATPRO
98	R-540.GATPRO Λ R-575.GATPRO → R-482.FLDIDT Λ R-568.FLDIDT
99	R-759.GATPRO → R-776.FLDIDT
100	R-789. Λ R-826. → R-861.
101	R-789. Λ R-861. → R-826.
102	R-826. Λ R-861. → R-789.
103	SURVEM-379.PRESDM → SURVM-108.PRESDM
104	SURVEM-379.PRESDM → SURVM-108.PRESDM Λ SURVM-461.PRESDM
105	SURVEM-379.PRESDM → SURVM-461.PRESDM
106	SURVEM-379.PRESDM Λ SURVM-108.PRESDM → SURVM-461.PRESDM
107	SURVEM-379.PRESDM Λ SURVM-461.PRESDM → SURVM-108.PRESDM
108	SURVEM-381.FLDRAW → SURVEM-381.VELSTK
109	SURVEM-381.FLDRAW → SURVM-591.FLDIDT
110	SURVEM-381.VELSTK → SURVEM-381.FLDRAW
111	SURVEM-578.ACQPD → SURVEM-578.FLDRAW Λ SURVEM-623.FLDIDT
112	SURVEM-578.ACQPD Λ SURVEM-578.FLDRAW → SURVEM-623.FLDIDT
113	SURVEM-578.ACQPD Λ SURVEM-623.FLDIDT → SURVEM-578.FLDRAW
114	SURVL-485.MISCEL → SURVL-485.PRESDM
115	SURVL-485.PRESDM → SURVL-485.MISCEL
116	SURVM-108.PRESDM Λ SURVR-784.FLDIDT → SURVM-461.PRESDM
117	SURVM-461.PRESDM Λ SURVR-784.FLDIDT → SURVM-108.PRESDM

	Continued 4 Significant Rules with Confidence $\geq 50\%$
118	SURVR-1045.STKNMO → SURVR-1592.GATPRO
119	SURVR-1592.VELSTK → SURVEM-346.FLDIDT
120	SURVR-482.FLDIDT Λ SURVR-568.FLDIDT → SURVR-540.GATPRO
121	SURVR-482.FLDIDT Λ SURVR-568.FLDIDT → SURVR-540.GATPRO Λ SURVR-575.GATPRO
122	SURVR-482.FLDIDT Λ SURVR-568.FLDIDT Λ SURVR-575.GATPRO → SURVR-540.GATPRO
123	SURVR-482.FLDIDT Λ SURVR-575.GATPRO → SURVR-540.GATPRO
124	SURVR-482.FLDIDT Λ SURVR-575.GATPRO → SURVR-540.GATPRO Λ SURVR-568.FLDIDT
125	SURVR-540.GATPRO Λ SURVR-568.FLDIDT → SURVR-482.FLDIDT
126	SURVR-540.GATPRO Λ SURVR-568.FLDIDT → SURVR-482.FLDIDT Λ SURVR-575.GATPRO
127	SURVR-540.GATPRO Λ SURVR-568.FLDIDT → SURVR-575.GATPRO
128	SURVR-540.GATPRO Λ SURVR-568.FLDIDT Λ SURVR-575.GATPRO → SURVR-482.FLDIDT
129	SURVR-540.GATPRO Λ SURVR-575.GATPRO → SURVR-482.FLDIDT
130	SURVR-540.GATPRO Λ SURVR-575.GATPRO → SURVR-482.FLDIDT Λ SURVR-568.FLDIDT
131	SURVR-540.GATPRO Λ SURVR-575.GATPRO → SURVR-568.FLDIDT
132	SURVR-568.FLDIDT Λ SURVR-575.GATPRO → SURVR-482.FLDIDT
133	SURVR-568.FLDIDT Λ SURVR-575.GATPRO → SURVR-540.GATPRO
134	SURVS-408.FLDRAW Λ SURVS-447.FLDRAW → SURVS-465.FLDRAW

Table D.1: Association Rules

Appendix E

Euclidian Distances

In this Appendix, it is explained why further generalisation of the association rules by means of the distance function is not appropriate. The following hypothesis is formulated

Sub-Hypothesis *The elements of the association rules are related to each other by a short distance to each other compared to distances between surveys in general.*

The Appendix is organised as follows. In section E.1 it is explained why the distance function is used. In section E.2, it is explained by means of several graphs that the distance is not short between the elements of rules compared to other distances.

E.1 Choice for Distance

The research database is constructed with the option for spatial objects. The table survey contains an attribute shape, which is filled with a polygon [20]. These polygons contain the feature that one can execute spatial queries on them.

E.1.1 Spatial Queries

The rationale for looking at the spatial relation between data sets is that data sets that are next to each are potentially accessed jointly. That is, seismic data acquisition was previously executed purely for exploration. That means that one does not know the location of hydrocarbons. Data was acquired in blocks. However, nature does not stay in those boxes, so oil and gas fields sometimes lie in two or even more boxes (e.g. the Groningen Gas Field contains more than 10 originally acquired boxes).

The first check was executed with the query *touches*. This query checks whether two polygons 'touch' each other. However, by manually checking this function, it was experienced that apparently touching polygons officially did not touch. The reason is that the coordinates are not precise enough.

Therefore, it is chosen to use the distance function, as other spatial queries also require very precise coordinates. The distance value shows the value "0" when polygons somewhere 'meet' each other (overlap, touches etc.).

E.2 Graphs

Figure E.2 shows the distances between all surveys, extracted from the research database. The meaning of this particular figure is that one expects that the distances between the elements of the rules will lie somewhere in the beginning (left) of the figure. So, this figure serves as comparison figure. Unfortunately, Figure E.2 reveals a distance distribution, which is hard to capture with statistical metrics. Therefore, the assessments must be executed by qualitative comparison between this Figure and the subsequent one.

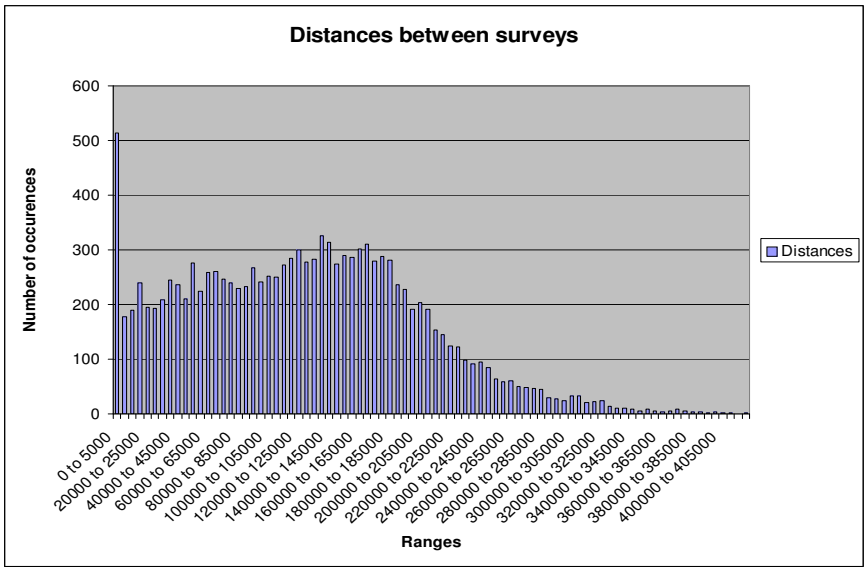

Figure E.1: Distance between all surveys

Figure E.2 shows the average distance between elements of the association rules. It must be noted that rules that are alike (so containing the same elements with respect to the survey, so the same distances) are regarded as one exemplar. Furthermore, the rules that contain elements from the same survey/directory

with a different data type are neglected here. The number of rules with that situation is 24. Unfortunately, for these rules further generalisation is not possible either (e.g. on the basis of data type).

One would expect that the most rules have small distances between the elements. However, it is seen in Figure E.2 that the distribution is comparable to the overall distribution of distances, both in format and in numbers (distance measures). That means that the distances between the rule elements are generally not shorter, compared to other distances. Therefore, it is concluded that the hypothesis is falsified.

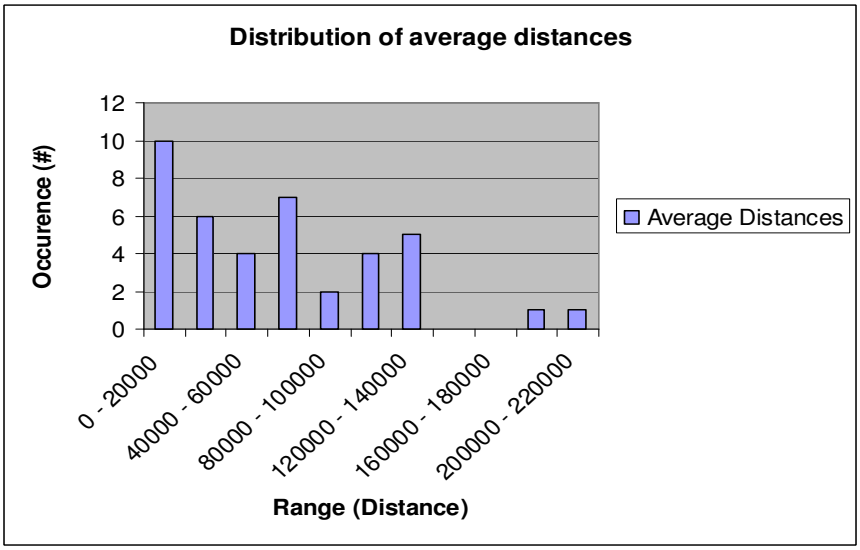

Figure E.2: Average distance rule elements

Appendix F

Visual Basic Code

In this appendix, the major elements of the Visual Basic Code is given of the two algorithms. Section F.1 contains the elements of the LRU algorithm simulation, whereas section F.2 contains the elements of the algorithm proposed by through the results of the data mining process.

F.1 The LRU algorithm

The LRU simulation consists of three major functions. The first one checks whether a combination of directory and file finds itself on a certain sheet (disk). Another function adds a file that is not yet present in front of the row. The last one takes a file from somewhere on the sheet and puts it in front.

F.1.1 Check existence

```
Function CheckExist(Row As Long) As Long

    ' Function CheckExist takes as input the Row from the Input Sheet. It returns a value. That value is
    ' either 10,000,000 (when the element is not present on disk) or the Row number in the Sheet that
    ' represents the content of the disk

    Dim Intermed As Long
    Dim DiskRow As Long
    Dim ValDir As String
    Dim ValFile As String

    Check = False
    DiskRow = 1

    ' Sheet 1 is the input disk, to which the following elements refer
    ValDir = Worksheets(1).Cells(Row, 4).Value
    ValFile = Worksheets(1).Cells(Row, 6).Value

    ' Walk through the Sheet that represents the disk and check whether the combination directory/file
    ' is present. Sheet 2 represents the disk content
    Do While Worksheets(2).Cells(DiskRow, 1) <> "" And Intermed = 0
        ' If it is present, extract the row number, which also stops the while loop
        If Worksheets(2).Cells(DiskRow, 1).Value = ValDir And Worksheets(2).Cells(DiskRow, 2).Value = ValFile
            Intermed = DiskRow
        End If
        DiskRow = DiskRow + 1
    Loop

    ' Write the Row number or the value 10,000,000 to the end value of Function CheckExist
    If Intermed = 0 Then
        CheckExist = 10000000
    Else:
        CheckExist = Intermed
    End If

End Function
```

Figure F.1: Function LRU CheckExist

F.1.2 Add file to disk or reorganize disk

The two functions that do something with the disk content. AddToRow adds a file in front of the row, whereas MutateRow reorganises and putts the just accessed file in front

```
Function AddToRow(OrigRow As Long, EndRow As Long) As Boolean

    ' This function puts a file (directory, filename and filesize) in front of a row
    ' As input it requires the row from the input (OrigRow) and the amount of rows on the disk sheet
    ' It moves everything one spot to the back, beginning with the last one

    Dim Column As Integer

    Do While EndRow > 1
        For Column = 1 To 3
            Worksheets("Disk").Cells(EndRow, Column).Value = Worksheets("Disk").Cells(EndRow - 1, Column).Value
        Next Column
        EndRow = EndRow - 1
    Loop
    Worksheets("Disk").Cells(1, 1).Value = Worksheets("Input").Cells(OrigRow, 4).Value
    Worksheets("Disk").Cells(1, 2).Value = Worksheets("Input").Cells(OrigRow, 6).Value
    Worksheets("Disk").Cells(1, 3).Value = Worksheets("Input").Cells(OrigRow, 7).Value

    AddToRow = True

End Function
Function MutateRow(OrigRow As Long, GoalRow As Long) As Boolean

    ' Function MutateRow equires as input the row on which the file is located.
    ' It grabs the file and puts it in front of the row

    Dim Column As Long

    For Column = 1 To 3
        Worksheets("Disk").Cells(1, Column + 5).Value = Worksheets("Disk").Cells(GoalRow, Column).Value
    Next Column
    Do While GoalRow > 1
        For Column = 1 To 3
            Worksheets("Disk").Cells(GoalRow, Column).Value = Worksheets("Disk").Cells(GoalRow - 1, Column).Value
        Next Column
        GoalRow = GoalRow - 1
    Loop
    For Column = 1 To 3
        Worksheets("Disk").Cells(1, Column).Value = Worksheets("Disk").Cells(1, Column + 5).Value
        Worksheets("Disk").Cells(1, Column + 5).Value = ""
    Next Column

    MutateRow = True

End Function
```

Figure F.2: Functions to add to or mutate the disk row

F.1.3 Main Macro

```
Sub Inpt()

' This is the Main Macro for this file. This Macro implements the LRU algorithm. It needs 3 sheets:
' On sheet 1, the input is represented, on sheet 2 the disk content and on 3 it is able to write the
' performance. The initial disk content is extracted from the database and consists of the last accessed
' (unique) files from the last year (2006)

Dim DRow As Long
Dim UseBool As Long
Dim InpRow As Long
Dim CheckRow As Long
Dim Size As Double
Dim UseRow As Long
Dim CheckEq As Boolean

' First count the number of rows ad the size of the initial disk
DRow = 1
Size = Size + Worksheets("Disk").Cells(DRow, 3).Value
Do While Worksheets("Disk").Cells(DRow + 1, 1) <> ""
    DRow = DRow + 1
    Size = Size + Worksheets("Disk").Cells(DRow + 1, 3).Value
Loop

' Then delete until the size finds itself under the threshold (80% of the disk size)
Do While Size > Worksheets("Parameters").Cells(2, 2).Value
    Size = Size - Worksheets("Disk").Cells(DRow, 3).Value
    For Column = 1 To 3
        Worksheets("Disk").Cells(DRow, Column).Value = ""
    Next Column
    DRow = DRow - 1
Loop

' Then start going through the input sheet
InpRow = 1
Do While Worksheets("Input").Cells(InpRow, 1).Value <> ""

    ' Use an extra parameter, as the row numbers will mutate during execution of the functions
     UseRow = DRow

     ' Use the function CheckExist to extract the existence and potentially the row on the disk sheet
     CheckRow = CheckExist(InpRow)

     ' If the file is not present
     If CheckRow = 10000000 Then

         ' If the activity is an access (opposed to an archive), then add 1 to the faults
         If Worksheets("Input").Cells(InpRow, 8).Value = "Acc" Then
             Worksheets("Perf").Cells(2, 1).Value = Worksheets("Perf").Cells(2, 1).Value + 1
         End If

         ' Add the size of the file to the total size and add the file in front of the disk sheet row
         Size = Size + Worksheets("Input").Cells(InpRow, 7).Value
         UseBool = AddToRow(InpRow, DRow + 1)
         UseRow = UseRow + 1

     ' But if the file is present
     Else:

         ' If the activity is an access (opposed to an archive), then add 1 to the hits
         ' Get the file from the row it is and out it in front of the row
         If Worksheets("Input").Cells(InpRow, 8).Value = "Acc" Then
             Worksheets("Perf").Cells(2, 2).Value = Worksheets("Perf").Cells(2, 2).Value + 1
             UseBool = MutateRow(InpRow, CheckRow)
         End If
     End If
     ' Again, delete the oldest files on disk until the total size is under the threshold
     If Size > Worksheets("Parameters").Cells(2, 2).Value Then
         Do While Size > Worksheets("Parameters").Cells(2, 2).Value
             Size = Size - Worksheets("Disk").Cells(UseRow, 3).Value
             For Column = 1 To 3
                 Worksheets("Disk").Cells(UseRow, Column).Value = ""
             Next Column
             UseRow = UseRow - 1
         Loop
     End If

     ' Set the right parameter for the number of rows in the disk sheet
     DRow = UseRow

     ' Go to the next input item
     InpRow = InpRow + 1
Loop

End Sub
```

Figure F.3: The Main Macro of the LRU Algorithm

F.2 DMP Algorithm

The algorithm consists of two phases. The first phase calculates the basic value. This is executed in advance. The second phase consists of walking through the input rows and 'prefetch' when appropriate according to the association rules. The calculation of the basic value is fairly straightforward and will, thus, not be included in this Appendix.

The prefetching algorithm on its turn consists of several functions, that will be given in this Appendix.

F.2.1 Main Macro

```
Sub Inpt()

    ' This is the main Macro of this algorithm. It uses several other functions, which are explained separately

    Dim Row As Long
    Dim UseBool As Boolean
    Dim RuleCheck As Boolean
    Dim ValTime As Double
    Dim NextTime As Boolean
    Dim PeriodRow As Long
    Dim RRow As Long
    Dim Num As Long
    Dim Trans As Long
    Dim Equal As Long
    Dim CacheSp As Double
    Dim CacheEnd As Boolean
    Dim ListRow As Long
    Dim SearchNeed As Boolean
    Dim Var1 As Double
    Dim Var2 As Double
    Dim EmptySeries As Long
    Dim VarRule As Long
    Dim Mean As Double
    Dim StDev As Double
    Dim CountCache As Long

    ' Set the startvalues on the right values
    CountCache = 0
    Var1 = 0
    Var2 = Var1
    Row = 1
    VarRule = 1
    PeriodRow = 1
    Trans = 1

    ' First go through the empty rows in the beginning
    Do While Worksheets(1).Cells(Row, 4) = ""
        Row = Row + 1
    Loop

    ' Walk through the rows until now transaction is left
    Do While Worksheets(1).Cells(Row, 1) <> ""

        SearchNeed = False

        ' If Column 4 is not empty there is a transactions
        If Worksheets(1).Cells(Row, 4) <> "" Then

            ' First Calculate the performance
            ValTime = Worksheets(1).Cells(Row, 2).Value

            ' Count the amount of files in the period
            Do While Worksheets(1).Cells(PeriodRow + Row, 2).Value = ValTime
                PeriodRow = PeriodRow + 1
            Loop

            ' The function Peformance runs through all the rows of the transaction
            UseBool = Performance(Row, Trans)

            ' The value of Row must be re-set
            Row = Row - PeriodRow

            ' And the transaction counter must be increased by 1
            Trans = Trans + 1

            ' Check whether it is relevant to diminish validities of files
            UseBool = CheckPrevSix(Row)

            ' If within 6 transactions (1,5 hour) there have been files accessed, the values must be diminished
            If UseBool = False Then
                UseBool = DiminishValidity(1)
            End If
```

Figure F.4: Data Mining based Algorithm Main Macro Part 1 of 3

```
UseBool = CountAndWrite(Row)
' Then delete equal rows and reorganize the rules list for all four lists
UseBool = DeleteId("RulesTempD")
UseBool = DeleteId("RulesTempS")
UseBool = DeleteId("RulesTempDT")
UseBool = DeleteId("RulesTempST")

' Re-set the row at the beginning of the period again
Row = Row - PeriodRow

' Set the validity of each accessed file to 6
UseBool = WriteValuePeriod(Row)

' Check the content of the relevant rules and update the variation value
' The rules are split in 4 seperate sheets of rules
RuleCheck = False
For Num = 18 To 21 '(all sheets with rules)
    RRow = 1
    If Num = 18 Then
    ' Walk through the list of rules and cath all potential rules
        Do While Worksheets("RulesTempD").Cells(RRow, 1).Value <> ""
            RuleCheck = RuleValCheck(RRow, "RulesTempD")
            If RuleCheck = True Then
                UseBool = UpdateVarRule(RRow, "RulesTempD")
                SearchNeed = True
            End If
            ' And delete the rule from the present rule list
            For Column = 1 To 6
                Worksheets(Num).Cells(RRow, Column).Value = ""
            Next Column
            ' To the next rule
            RRow = RRow + 1
        Loop
    Else: '(For all the rule types (4 in total) the same process)
    ' .....
    End If
Next Num
' Check whether there are files present with a variative value
If Not (SearchNeed = False And Var1 = 0) Then
    ' New variative value (all together)
    Var2 = CountTotalVar
End If

' If the variative value differs from the previous transaction
If Var1 <> Var2 Then
    Var1 = Var2

    ' A new caching candidate list must be constructed
    ' First the old list must be deleted
    UseBool = DeleteList("List")

    ' The Mean and Standard Deviation of the Value must be altered
    StDev = StDevVal

    ' If the total variative value is 0, then the basic list can be used (only basic values incorporated)
    If Var2 = 0 Then
        ExtraRow = 1
            Do While Worksheets("BasicList").Cells(ExtraRow, 1) <> ""
                For Column = 1 To 4
                    Worksheets("List").Cells(ExtraRow, Column).Value = Worksheets("BasicList").Cells(ExtraRow,Column)
                Next Column
            Loop
```

Figure F.5: Data Mining based Algorithm Main Macro Part 2 of 3

```
            Else:
                ' Otherwise, a new candidate list must be constructed
                UseBool = MakeUpList

                ' Then the cache must be updated
                ' From the top of the potential list, it is checked whether the file is already present
                ' in the cache. Constraint for further caching can be two things: the list with candidates
                ' has finished or the writing resources have been used

                ' The available writing resources is
                CacheSp = 0
                CacheSp = CalculateCacheSpace(Row)

                ' Check the list on presence in the cache and compare value
                ' If relevant, put in the cache and delete the files with the least values of the cache
                ' Give to the function the available Caching resources so that it will not exceed the cache space
                UseBool = ToCache(CacheSp)
            End If
        End If
    Else:
        ' If no writing or archiving is exected only the validity of the values must be diminishes
        ' This can be done for several rows in once if applicable to save overhead costs of the algorithm
        EmptySeries = 1
        Do While Worksheets(1).Cells(Row + EmptySeries, 4) = ""
            EmptySeries = EmptySeries + 1
        Loop
        Row = Row + EmptySeries - 1
        UseBool = DiminishValidity(EmptySeries)
    End If

    ' Go to the next transaction
    Row = Row + PeriodRow
    PeriodRow = 1

    Loop

End Sub
```

Figure F.6: Data Mining based Algorithm Main Macro Part 3 of 3

F.2.2 Functions

In the main Macro, one can find the use of several functions. All together, 16 functions are required to execute the main Macro. The most important of course is the function that writes the performance of the algorithm per transaction. The remaining functions are not given in this thesis. It is assumed that the comments are sufficient to understand what the functions do.

```
Function Performance(Row As Long, EndRow As Long) As Boolean

    ' This function calculates the performance in terms of hitrate
    ' It compares the files from the input with those in the cache

    Dim DateVal As Long
    Dim TimeVal As Double
    Dim Success As Long
    Dim Counter As Long
    Dim Succ As Boolean
    Dim CacheRow As Long

    ' Set the start parameters on the right values
    Counter = 0
    Success = 0
    Succ = False
    DateVal = Worksheets(1).Cells(Row, 1).Value
    TimeVal = Worksheets(1).Cells(Row, 2).Value

    ' While the period of the transaction is not yet over
    Do While Worksheets(1).Cells(Row, 1).Value = DateVal And Worksheets(1).Cells(Row, 2).Value = TimeVal

        ' And the element of the transaction is an access
        If Worksheets(1).Cells(Row, 9).Value = "Acc" Then
            ' The total amount goes 1 up
            Counter = Counter + 1

            ' It is checked whether the file finds itself in the cache
            CacheRow = 1
            Do While Worksheets("Cache").Cells(CacheRow, 1) <> "" And Succ = False
                ' If so....
                If Worksheets(1).Cells(Row, 5).Value = Worksheets("Cache").Cells(CacheRow, 1).Value And Worksheets(1)
                    ' The number of successes goes 1 up
                    Success = Success + 1
                    Succ = True
                End If
                CacheRow = CacheRow + 1
             Loop
             Succ = False
         End If
         Row = Row + 1
    Loop

    ' The performance values are written in terms of faults and hits
    Worksheets("Performance").Cells(EndRow, 1).Value = EndRow - 1
    Worksheets("Performance").Cells(EndRow, 2).Value = Success
    Worksheets("Performance").Cells(EndRow, 3).Value = Counter - Success

    Performance = True
End Function
```

Figure F.7: Performance Calculation of the Data mining results driven agorithm

List of Figures

List of Tables