University of Economics in Prague

Faculty of Informatics and Statistics

Department of Information and Knowledge Engineering

Specialization: Knowledge Technologies

Master Thesis

Clickstream Analysis

Author: Bc. Tomáš Kliegr

Supervisor: Doc. RNDr. Jan Rauch, CSc.

Printed on recycled paper

Typed with TEX

Prague 2007

Declaration

I herewith declare that I have produced the master thesis "Clickstream Analysis" and the accompanying CD independently and without using any other aids than listed in the Bibliography. Any thoughts directly or indirectly taken from somebody elses sources are made discernible as such.

In Prague, July 10, 2007 Signature

to my father

Acknowledgments

I am indebted to a number of people who contributed to this master thesis. The patience, encouragement and support of my supervisor Doc. Jan Rauch made it a pleasure to write. I am also grateful to the valuable comments and feedback which I obtained from the members the Knowledge Engineering Group of UEP and the Personalized Web group. Special thanks goes to Vojtěch Svátek for his advice on the semantic side of web usage mining and for his commitment to organizing the highly inspirative KEG seminars.

I have been lucky to have been granted a one semester scholarship by the Slovak Ministry of Education which allowed me to spend one semester at the Slovak Technical University in Bratislava under the supervision of Prof. Mária Bieliková. This stay has significantly contributed to the practical part of this thesis. I am especially grateful to Dr. Gabriela Kosková for the lectures on Data mining and for her feedback on the seminar work, which became the basis of the first part of Chapter 4. I am equally grateful to Prof. Jiří Pospíchal for his highly interesting lectures on Evolutionary algorithms, challenging seminars and for his valuable comments on the seminar work, which is the groundwork of the second part of Chapter 4.

Gratitude must be expressed to the the Internal Grant Agency of the UEP and Grant Agency of the Czech Republic for their support of my attendance on scientific conferences Znalosti 2006, EKAW 2006 and Znalosti 2007, which have helped to broaden my horizons.

Finally and most importantly, I would like to thank my relatives and my girlfriend for their support.

Abstract

The aim of this thesis was to introduce current research trends in clickstream analysis and to propose a new heuristic that could be used for dimensionality reduction of semantically enriched data in Web Usage Mining. Generally, the application of Web Usage Mining (WUM) and Semantic Web technologies in clickstream analysis of E-commerce websites is in the focus of this thesis.

The following phases of the mining process (according to CRISP-DM methodology) are covered: Business understanding, Data acquisition, Data preprocessing and Data modeling. Existing experimental approaches which the author considered as the most viable for use in an e-commerce scenario are presented. Click-fraud and conversion fraud are identified as key prospective application areas for WUM, which is illustrated by documenting a previously unpublished conversion fraud attack against a widely used clickstream analysis solution (Google Analytics). A proposed defense is a robust solution for collecting semantically enriched clickstream data dubbed Google Analytics INterceptor (GAIN). This solution is to the best of the author's knowledge first third party "plug-in" for Google Analytics. Various ways of clickstream collection are described and compared as well, including the acquisition of semantics.

Web usage data, especially when semantically enriched, tend to have high dimensionality. Existing approaches to dimensionality reduction are reviewed and a new heuristic for determining the main interest of the visitor is proposed along with several new path summarization attributes. These attributes replace the high-dimensional variable-length clickstream in the data preprocessing step. Its output, the visitor profile, serves as input for datamining algorithms. Real world visitor profiles are used to assess the performance of two association rule mining systems 4ft-Miner (GUHA algorithm) and Arules (Apriori) and three variants of K-means clustering algorithm. It is concluded that 4ft-Miner provides better functionality with comparable time-complexity. Spherical K-means, the simplest algorithm in the experiment, is recommended for clustering.

Abstrakt

Cílem této práce je představit soudobé výzkumné trendy v oblasti click stream analýzy a navrhnout novou heuristiku, která by mohla být užita pro snížení dimenzionality sémanticky obohacených dat používaných ve Web Usage Miningu (WUM). Aplikace technologií WUM a Sémantického webu v oblasti clickstream analýzy je obecně v ohnisku zájmu této práce.

V práci jsou popsané následující fáze procesu dobývání znalostí z databází (dle CRISP-DM): Porozumění problematice, Porozumění datům, Příprava dat, Modelování. Prezentovány jsou ty experimentální přístupy, které autor považoval za nejlépe použitelné na webech spadajících pod E-commerce. Podvodná kliknutí a podvodné konverze byly identifikovány jako potenciální oblasti pro aplikaci WUM, což je i ilustrováno poprvé zdokumentovaným útokem typu "podvodná konverze" proti široce používanému systému pro analýzu clickstreamů (proudů kliknutí) Google Analytics. Navrhovanou obranou proti tomuto útoku je nové robustní řešení pro sběr a ukládání sémanticky obohacených clickstreamů nazvané Google Analytics INterceptor (GAIN). Tento software je, pokud je autorovi této práce známo, první "plug-in" pro Google Analytics. V práci je popsána a porovnána řada dalších způsobů pro sběr dat pro WUM, včetně získání sémantiky.

Data týkající se používání webu mívají velmi vysokou dimenzi, zvláště pak tehdy pokud byla sémanticky obohacena. Nejdříve je podána stručná charakteristika existujících přístupů k snížení dimenzionality, a poté je navržena nová heuristika pro vytvoření nových atributů obsahujících odhad zájmu návštěvníka webu a několik dalších nových atributů shrnujících pohyb návštěvníka po webu. Během předzpracování jsou tyto atributy použity k nahrazení příliš mnoha rozměrného clickstreamu, jehož dimenze je navíc proměnlivá. Výstupem předzpracování je profil návštěvníka, který již slouží jako vstup pro algoritmy dobývání znalostí z databází. Profily návštěvníků zkonstruované na základě reálných dat jsou použity při hodnocení dvou systémů pro analýzu asociačních pravidel, a to procedury 4ft-Miner (algoritmus GUHA) ze systému LISp-Miner a balíčku Arules (algoritmus Apriori) z prostředí R, a třech variant shlukovacího algoritmu K-means. Závěrem je doporučen 4ft-Miner, který nabízí lepší funkcionalitu při časové složitosti empiricky porovnatelné s Arules. Sférický K-means, nejjednodušší z porovnávaných algoritmů, je doporučen pro shlukovací úlohy.

Contents

List of Figures

List of Tables

List of Abbreviations

AAI Above Average Implication (LISp-Miner quantifier)

AIMP Absolute difference of improvement to 1 (Arules interest measure)

CPC Cost Per Click (advertising scheme)

CPA Cost Per Action (advertising scheme)

DoC Degree of Coherence (measure of visual coherence of a block)

GA Google Analytics (clickstream analytics sofware)

GAIN Google Analytics INterceptor (Google Analytics plug-in)

GP Genetic Programming

ISP Internet Service Provider

LHS Left-Hand Side of an association rule

LM LISp-Miner (data mining software)

OPA On Page Attribute (semantic extension of Google Analytics exchange format)

PPC Pay Per Click (advertising scheme) same as CPC

RHS Right-Hand Side of an association rule

SERP Search Engine Result Page

UAC User Activity Collector

VUM Vision-based Usage Mining server

WUM Web Usage Mining

Introduction

The advent of e-commerce has meant a dramatic shift in the contact point between the business and its customers. Even a small enterprise can no longer rely entirely on the information obtained during the eye-to-eye contact with its clients. Many companies quickly realized that if they want to gain insight into the needs, interests and wishes of the visitors of their websites, they have to utilize some kind of a web monitoring tool. According to a recent survey [33], most of these tools are based on statistical analysis, while only a few of them actually exploit data mining techniques.

While utilizing *some* web monitoring tool has become a de facto standard in e-commerce, a company can still gain a significant competitive advantage by using an advanced – data mining based – tool to analyze its web site traffic e.g. to increase the conversion rate and detect click-fraud. Data mining is defined as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data"[36] or "the science of extracting useful information from large data sets or databases"[46].

The application of data mining techniques to discover patterns from the Web is called Web Mining. Web Mining could be further defined as

> the discovery of and analysis of useful information from the World Wide Web. This describes the automatic search of information resource available online, i.e., Web content mining, and the discovery of user access patterns from Web services, i.e., Web usage mining (Cooley, Mobasher, Srivastava [29]).

Web Usage Mining (WUM) is a term used in the scientific community for data mining applications in the discovery and analysis of interesting patterns in the web site usage data. A related term, clickstream analysis, is more popular among e-commerce practitioners. Clicks stream analysis could be defined[1] as

> the process of collecting, analyzing and reporting aggregate data about which pages visitors visit in what order - which are the result of the succession of mouse clicks each visitor makes (that is, the clickstream). There are two levels of clickstream analysis, traffic analysis and e-commerce analysis. Traffic analysis operates at the server level by collecting clickstream data related to the path the user takes when navigating through the site. Traffic analysis tracks how many pages are served to the user, how long it takes pages to load, how often the user hits the browser's back or stop button, and how much data is transmitted before a user moves on. E-commerce-based analysis uses clickstream data to determine the effectiveness of the site as a channel-to-market by quantifying the user's behavior while on the Web site.

[1] `http://www.bitpipe.com/tlist/Web-Analytics.html`

Explanation Of Thesis Title

The reason why this thesis is entitled clickstream analysis and not Web usage mining is two fold. First reason is very pragmatic - it encompasses not only the actual analysis or mining of the data, but the complete data analysis life cycle including data collection, which is according to the aforementioned definition a clickstream analysis task.

The second reason is rather symbolic. Clickstream analysis is a practical, comprehensive day-to-day task of an e-powered business, whose tools have still a little overlap with the mostly theoretical work done under the hood of Web Usage Mining. The primary goal of the author of this thesis was to hint how the clickstream analysis routine could be redesigned with the use of advanced algorithms from the Web Usage Mining toolbox[2].

Thesis Goal

The ultimate goal of this thesis is to present the advances in Web Usage Mining from a practical perspective. The intermediate goals were to a) summarize the available literature on this topic and b) to propose a solution to one of the pressing problems in this field. These goals were to be addressed holistically, following the CRISP-DM methodology when possible. The thesis assignment gave an option between addressing the following two problems:

1. Suggest a heuristic that would be able to identify the main interest of the visitor, set its parameters for a given website and test it.

2. Suggest a heuristic for determining the relevancy of results returned by an Internet Search engine for a given website.

Both these problems are addressed in this thesis, however not entirely.

The theory for the heuristic which would be able to solve the Problem 1 was finished and it is available in Chapter 3.7. Unfortunately, this heuristic relies on a provision of semantically enriched data and a training set, which are difficult to obtain with existing tools. The author therefore opted to design a new clickstream collection software that allows to gather both the semantically enriched clickstreams and the training set. This software dubbed Google Analytics INterceptor (GAIN) works as a kind of plug-in for the widely used Google Analytics clickstream analysis software. Insufficient amount of time was left to use this software to obtain the necessary training set.

To make-up for the unsolved part of the Problem 1, the author implemented three variations of the K-means clustering algorithm (see Sections 4.8, 4.7, 4.6) and tested them on real-world data (see Section 7.4) in order to cluster visitor's of a tour operator's website according to their interest in trips. The implementation introduced in Section 4.8 is a novel one; it uses a genetic algorithm for initialization.

Problem 2 was addressed in Section 3.6. In addition to the proposed heuristic for determining the relevancy of Search Engine results, several other new attributes are proposed as well.

[2]This is not to say that there are no clickstream analysis solutions, which utilize data mining techniques. The market with Web Mining software is thriving, between 2000 and 2003 the number of existing products has nearly doubled [33]. According to the same source, the absolute number of these products is still,however, rather low. The current list of web usage mining and web mining software is available at `http://www.kdnuggets.com/software/web-mining.html`

Contribution And Thesis Organization

The contribution of this thesis beyond the aforementioned design of the heuristics is spread among nearly all phases of the data analysis process as specified by the CRISP-DM Methodology [24]. In fact, the thesis structure loosely corresponds to two runs of the CRISP-DM cycle. The first descriptive part of the thesis focuses on chosen aspects of the individual steps which need to be taken during clickstream analysis. The second part is case-study driven. It shows selected problems presented in the theoretical part in real world situations and shows how selected algorithms introduced in the theoretical part perform on real-world data.

The theoretical part of the thesis is opened with a chapter on click and conversion fraud, which is one of the most promising application fields for WUM-based clickstream analysis. This chapter gives almost exhaustive review of the scientific literature on this topic, which is critically assessed with respect to possible applications of data mining algorithms.

The three following chapters bring a selection of existing algorithms and techniques for collection, preparation and analysis of web usage data. Section 2.1 of the Data Acquisition Chapter focuses on collection of clickstreams. It compares the quality of data obtained by client-side and server-side tracking, reasons to use the former are given. Section 2.2 deals with the acquisition of metadata. Metadata in Web Usage Mining are often neglected, perhaps because of their supposedly high costs. This sections discusses which of the several alternatives to acquisition and storage of semantics are the most suitable in an e-commerce scenario and which kind of metadata have the most favourable cost/benefit ratio.

The contribution of the theoretical part of the thesis culminates in Chapter 3, which is focused mainly on solving the dimensionality reduction problem in a WUM scenario. Sections 3.1 – 3.3 formulate the problem and introduce the *curse of dimensionality*. Existing solutions to dimensionality reduction are reviewed in Sections 3.4 – 3.5. In the following two sections, the thesis author extends his work on the data-preprocessing algorithm for dimensionality reduction first introduced in his bachelor thesis [53]. In Section 3.6 several new WUM-specific attributes are proposed that could be used for dimensionality reduction. In Section 3.7 the author suggests a new heuristic for determining the main interest of the visitor.

Chapter 4 presents some implementations of selected datamining software and algorithms. Section 4.1 – 4.3 presents implementations of two association rule mining algorithms – *GUHA* and *Apriori*. These two algorithms are compared with respect to their selected implementations, *4ft-Miner* procedure of the *LISp-Miner* system (GUHA) and *Arule* package of the *R* data mining environment (apriori). Interest measures can significantly facilitate the generation of association rules or their postprocessing. Section 4.4 shows that choosing the right interest measure is not a trivial task and gives some hints to aid this process. Clustering is an important data-mining technique often used in WUM, an in-depth description of three variants of the K-means algorithm as implemented by the thesis author is presented in Sections 4.5–4.8. Section 4.8 gives account of the author's attempt to improve the clustering results of a hierarchical variation of K-means called Repeated Bisection K-means presented in Section 4.7 by using evolutionary algorithm for initialization. The chapter is concluded by Section 4.9 with some remarks on choosing the appropriate data mining solution.

Chapter 5, the first chapter of the practical part of the thesis, is motivational for WUM as it discloses an attack against a widely used clickstream analysis solution *Google Analytics*. As far as the author knows, this attack has not been documented before. The author tried to identify possible security weaknesses of the Google Analytics. This task required to reverse engineer the Urchin Sensor module, which is a part of Google Analytics that collects the

data. Section 5.1 presents the Google Analytics exchange format, which is to the best of the authors knowledge its first publicly available description. A proof of concept attack is carried out on a real website in Section 5.2. Defense against the attack is proposed in Section 5.3.

A robust solution dubbed GAIN for collecting semantically enriched clickstream data proposed and implemented by thesis author is introduced in Chapter 6. GAIN is intended to complement Google Analytics and provide a website which uses Google Analytics with increased possibilities for fraud detection as well as with sufficiently granular web usage data to allow for data mining.

In Chapter 7, the final chapter of this thesis, all algorithms introduced in Chapter 4 are run on the same real world web usage dataset in order to evaluate the individual algorithms and demonstrate their usability in practice. The dataset used was semantically enriched as described in Section 7.1. Some of the preprocessing techniques presented in Chapter 3 are demonstrated in Section 7.2. The experiments with the two association rule mining systems *LISp-Miner* and *Arules* including a performance benchmark of the two systems are described in Section 7.3. This is judging from the available literature one of few comparisons of GUHA and Apriori algorithm implementations. The three variants of the popular clustering K-means algorithm are described in Section 7.4. The usability of data mining results in practice is evaluated in Chapter 7.5.

The conclusion evaluates how the goals of the thesis were fulfilled and makes proposals for further work.

Examples

The birth of this thesis was case study driven. The author has tried to identify the pieces of theory that could be applied to the advantage of a real world e-commerce website of a Czech tour operator CK Poznání. This company has kindly allowed the author to deploy his own software for collecting data from its website and use this data for experiments presented in Chapter 7. The GAIN software (see Section 6) was also tested on this website; the choice of the technologies used for its implementation was significantly influenced by its this first deployment. In order to give the thesis a more real-world feel, most examples used throughout it are either directly based on real-world data from this website, or at least inspired by it.

Prerequisites

This thesis is not an introduction to clickstream analysis, data mining algorithms or information retrieval. The author tries to briefly introduce most of the concepts from these areas when they are used for the first time, but a thorough explanation is in many cases beyond the scope of this thesis. Further information on these topics can be found in the following excellent publications, which were also extensively used as reference guides by the author: *Web Metrics* by Jim Sterne [73], *Data Mining: Practical Machine Learning Tools and Techniques* by Ian Witten [86] and the *Introduction to Modern Information Retrieval* by Gerard Salton [68]. In addition, a background in web engineering is expected from the reader and hence no explanation is usually given for web-engineering related terms such as robot, web spider or cookie. Please refer to any web-engineering text-book for further definition if needed.

Part I

Theory

Chapter 1

Conversion Fraud And Its Detection In PPC And PPA Advertising

Almost all e-commerce businesses now rely on efficient on-line advertising. Web Usage Mining can play an important role in achieving this goal. It can help to choose which advertisings bring the best price/benefit ratio and it can help to disclose fraud. Fraudulent behaviour flourishes on the Internet, because of its anonymity, global character and poor law enforcement.

Click- and conversion-fraud is a serious threat to most e-commerce businesses that advertise on-line. The goal of this chapter is to present various ways of the defense against this kind of fraud and to show that among all the various approaches to it, data-mining algorithm have a strong place.

First, it is discussed whether this kinds of fraud cannot be prevented by using a fraud-resistant advertising model. The widely held assumption that the pay-per-action (PPA) model is less susceptible to fraud than pay-per-click (abbreviated as PPC sometimes as CPC) model will be challenged. Some of the more sophisticated attacks against PPC are discussed together with possibilities to detect these attacks or remove incentives for them. For the research on fraud in PPA is scarce, we present several modifications of the known PPC attacks and discuss their overall feasibility and detectability. The ultimate goal of this chapter is to point attention to a promising application field for Web Usage Mining algorithms and also discuss the limitation of these algorithms and alternatives. In fact, it corresponds to the Business Understanding phase of the CRISP-DM Methodology.

The ideas presented in this chapter are further developed in the case study part of this thesis. In Chapter 5 it is shown, how Google Analytics, current mainstream hosted web analytics software, collects data and ensures their validity and how an over-reliance on only one solution can lead to easy susceptibility to conversion fraud. A proof-of-concept attack is discussed there as well. In Chapter 6 a data collection system GAIN is proposed to gather data in parallel with Google Analytics in order to increase the fraud detection possibilities.

This chapter is organized as follows. Because the research on Conversion fraud is scarce, Section 1.1 gives account of existing research on click fraud. Section 1.2 briefly introduces the pay-per-action (PPA) model and especially the possible definitions of actions (conversions[1]).

[1]In further text, we will often use the terms *action* and *conversion* interchangeably, because the notion of

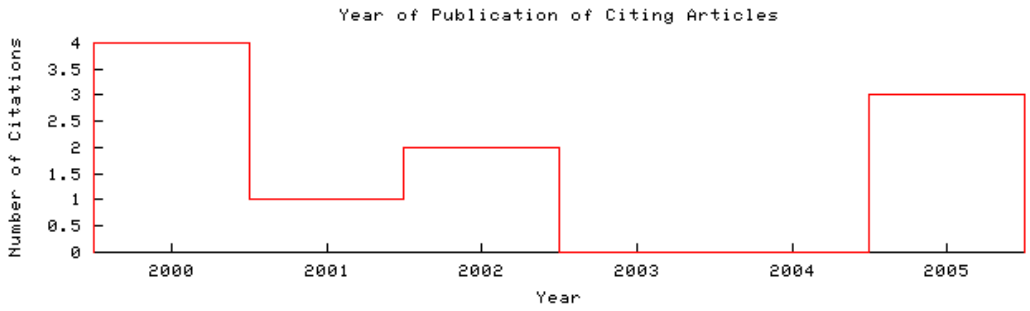


Figure 1.1: Citation distribution for On the security of pay-per-click and other Web advertising schemes. Source: citeseer.ist.psu.edu

These definitions are important for the discourse on the possible attacks against PPA in section 1.3.

1.1 Related Research

When Google was used to find documents matching "Click Fraud" query on Citeseer, only one document was found. Although this one document is of course not the only academic paper dealing with click fraud, this well illustrates the fact that research on click fraud is not yet abundant. A lack of academic papers dealing with this subject is also reflected in the References section of the existing papers – only relatively handful of references is listed and out of these an unusually large portion are papers published only on web sites or even in general subject newspapers like Wall Street Journal. If we consider implication of this according to the Bradford's law [21], it seems as if there was not yet any common platform for papers on click fraud.

A milestone paper on click fraud was, in the author's opinion, "On the security of pay-per-click and other Web advertising schemes" [12] published in 1999. The graph 1.1 depicting the distribution of citations of this paper in time shows a resurgence of interest in click fraud. Although the paper is quite old and should be subject to information aging and consequencing exponential drop in the number of citation over time, there are no symptoms of this happening.

1.1.1 Hit Inflation Attack

In paper [12], Anupam et al. show a hit inflation attack , which "is virtually impossible for the program provider to detect conclusively". In a hit inflation attack, an advertiser A pays a partner publishing website P a certain amount for every visitor coming from P to A. Partner website P is dishonest and makes an alliance with another dishonest website S. Website S is a relatively busy website with many visitors.

The attack is based on an ingenious method, which makes some visitors of S request a page from P and then A without the visitors noticing. Moreover, the Referrer fields included in their HTTP request indicate to A that these visitors came from P. The wickedness of this

action in PPA model is narrowed to conversion. It is clear that *action* has generally wider meaning. It could be argued that more fitting name for the Pay-Per-Action would be Pay-Per-Conversion. The reason why the former name was adopted probably was that the abbreviation of Pay-Per-Conversion would be the same as the existing abbreviation for Pay-Per-Click(PPC), which would be confusing.

attack stems from the fact that the fraudulent requests made to site A come from real users with unique and uncompromised IPs using a range of browsers and coming in random time spans.

1.1.2 Detecting Fraud With Association Rules

The most recent (according to Citeseer) attempt to detect this "undetectable attack" is paper "Using Association Rules for Fraud Detection in Web Advertising Networks" by Agrawal et al [57]. In this paper, the authors claim that this kind of fraud had not been successfully detected by previously existing techniques and propose a novel algorithm that should be able to detect it. Their algorithm *Streaming-Rules* is a novel improvement of association rules concept which introduces a new notion of patterns and methods for their identification in data streams. The motivation behind their approach is a claim that

> The only entity that can detect the association between the dishonest publisher, P, and the dishonest Web site, S, is the Internet Service Provider (ISP), through which the customer logs on to the Internet.

The authors ran their algorithm on real-world data – anonymized streams of HTTP requests from an ISP. They observed that their algorithm found a set of suspicious sites A that were always being requested before another set of sites B with confidence at least 0.5. They conclude that it was not possible to check out the results even if they knew the URLs. They gave the following rationale:

> Internet Browsers record the referring site, and not the referred to site, in the history. For instance, if page a.com has an invisible frame of height 0, where another page b.net gets loaded when a loads, the Browser records only a.com in the history. Thus, visiting a.com, and then looking for b.net in the history is not effective. Only an entity that monitors the HTTP requests made, like ISPs, can test a relationship that Streaming-Rules reported for being fraudulent.

Although this technique for detecting click fraud is very interesting and would probably be very effective if the data from the ISPs were available, we think that this method can not be utilized in practice on a wide scale. In our criticism of the approach, we do not question any of the five "Assumptions in Modeling the Problem" that the author state (refer to [57]), but another, tacit, assumption – the availability of the data from ISPs.

Due to the global and decentralized nature of the Internet and the number of ISPs on the market, the assumption of having access to a large portion of the Internet traffic is highly unrealistic. Moreover, even if the data were available, the computational costs would likely be to high. There is still the possibility of sampling, obtaining and processing only data from only few ISPs as the authors did in the paper. Although the samples could contain some fraudulent behaviour, which could be disclosed by *Streaming-Rules* algorithm, it is quite unlikely that the sample from several ISPs would contain transactions pertaining to a specific fraudster P and specific advertiser A.

Dishonest publishers (or their allies S) are unlikely to be very large sites as those would probably not risk loosing the trust of advertisers by employing dishonest practices to earn a small premium to their relatively large honest earnings. The Streaming-Rules algorithm rules would thus be in our opinion viable, only if it were carried out by some "Internet Police"

rather than by an individual advertiser who suspects being subject to the hit-inflation attack introduced in [12].

1.1.3 Reverting To A (Modified) Pay Per Impression Scheme

Another way how to solve the "undetectable" attack was described by Goodman in 2005 [40]. He proposes to complement the PPC model with pay-per-percentage of impressions scheme, which should be able to circumvent rather than detect this kind of attack. In this model, advertisers pay publishers for a percentage of impressions during which their ad occurs rather than for individual click-throughs. The weakness of the original pay-per-impression scheme was the susceptibility to impression-inflation attack. A dishonest competitor could use fake impressions to exhaust the budget of the advertiser by fake searches (when the ads are placed on a search engine) or fake visits (when it is on the site of a publisher).

The pay-per-percentage of impressions scheme effectively eliminates this kind of fraud, because the advertiser pays for a percentage of impressions the publisher receives in a certain time period. The price for the percentage of impressions is settled based on the volume of traffic the site obtained in the past. If the traffic on the publisher's site increases due to fake visits, neither the amount paid nor the number of real impression change for the advertiser.

Unfortunately, this method has many disadvantages. As the author notes, it is not suitable for affiliate advertising. There are open-questions pertaining to the pricing of the pay-per-percentage-of-impressions advertising. A third party authority would be needed to independently report on the traffic the publisher's site receives. Moreover, for a single page, the volume of past traffic might be insufficient. The fact that this technique is not generally usable in affiliate advertising is unfortunate as most fraud happens exactly in affiliate advertising (such as Google AdSense). The publishers have a straightforward motivation – fraud results in immediate monetary gain. This is a more straightforward motivation rather than causing a loss to a competitor as in search advertising (such as Google AdWords).

Even for search engines, the pay-per-percentage-of-impressions might be subject to a new kind of fraud, which Goodman coined "Misinformation Fraud". This fraud is feasible for not very popular but highly competitive search terms (e.g. terms related to luxurious goods). For example, consider keyword which obtains 500 hits per search period. A dishonest competitor could generated 100 false searches and thus increase the value of one percent of impressions for the advertiser. The advertiser would be willing to pay 1$ per impression. The honest advertisers seeing that the popularity of this search term increased would also increase their bids, thus driving the price per one percentage of impressions higher from 5$ to 6$. Although the per-impression price would stay 1$, the price per real impression would rise to 1.2$. The honest advertisers would loose 100$ from their advertising budget; the dishonest competitor would in the meantime spend her advertising budget on other relevant but cheaper keywords.

The pay-per-percentage of conversions method was criticized by Immorlica et al [48]. They give the list of following additional disadvantages of pay-per-impression model:

- Advertisers are used to buying clicks rather than impressions.
- By selling clicks, the service provider subsumes some of the risk on the marketplace due to natural fluctuations.
- Estimating worth of an impression is more difficult for advertisers than estimating worth of a click.

1.1.4 Click-Per-Conversion Schemes Resistant To Attacks

The authors [48] propose their own method for tackling click-fraud, which is based on pay-per-click model. They introduce a new way of estimating the click-through rate (CTR) of an advertisement. The CTR is a probability that an impression will lead to a click. Each advertiser bids how much is she willing to pay for a click. The advertisers are then ranked according to the product of their bids and their CTRs. Their advertisements are then displayed in the order of their ranks. The price the advertiser pays for one click decreases with increasing CTR.

As a result, a hit-inflation attack carried out against an advertiser has two opposing effects: obviously, the advertiser has to pay for the fraudulent click (short-term loss), however, as his CTR increases the price of future clicks decreases (long-term benefit). The basic intention of [48] was to find such definition of CTR, which would be resistant to attack. The authors prove that a particular class of CTR learning algorithms (*click-based* algorithms) have the property that the short-term loss and long-term benefit cancel out.

In paper [48] it is claimed that most service providers now detect fraudulent clicks with machine learning approaches and that these approaches are ineffective: "Recent tricks, like using cheap labor in India to generate these fraudulent clicks, make it virtually impossible to use these machine learning algorithms". The first-best approach is to remove incentives for click-fraud rather than detect it.

Following concerns about the level of click-fraud on Google, Google hired an independent expert Alexander Tuzhilin to investigate the quality of its practices on disclosing and preventing click fraud [81]. According to this report, Google, a major service provider, does not use machine learning techniques (classifier-based) except for only "a couple of relatively minor cases." In Google, they use rule-based and anomaly based detection techniques, which are used both on-line and off-line.

The on-line click fraud detection in Google comprises of a collection of filters. Tuzhilin notes that the structure of "most of Googles filters, with a few exceptions, is surprisingly simple." These filters reportedly work reasonably well even against more sophisticated attacks due to the following factors:

1. Combination of filters.
2. Extra complexity of some of the filters.
3. Simplicity of most of the attacks.
4. The Long Tail of invalid clicks.

Although Tuzhilin agrees with the satisfactory performance of Google filters, he warns that this might change as the attacks will shift toward the Long Tail on Figure 1.2. According to a hypothesis offered by Tuzhilin, Google filters perform reasonably well, because the majority of attacks are relatively simple ones as the distribution of attacks is zipfian.

Tuzhilin also notes that he does not have any scientific evidence for this distribution to be zipfian.

1.1.5 Hit Shaving

Advertising schemes can not only be abused by the publisher or evil competitor, but also by the advertiser, who can claim to obtain less impressions than in reality in an effort to lower

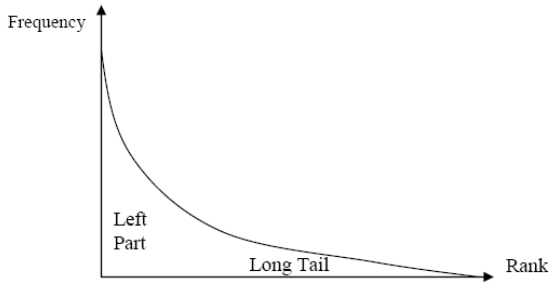


Figure 1.2: The intricacy vs frequency of attacks follows Zipf's distribution. Source: [81]

payments to the publisher. This attack is called hit shaving. Reiter et al. [67] recommend a range of useful techniques publishers could use to monitor the number of click-throughs for which they should be payed. The proposed techniques span from a simple redirection script which records every click from the Publisher to the Advertiser. Links from content pages of the publisher to the advertiser are not direct. Instead, a redirection script placed either on the publishers website or on a more trusted third-party site is a target of these links. The redirection script records the click and forwards invisibly the visitor's browser to the advertisers site.

This simple technique is often used in practice. The authors also discuss the use of more sophisticated techniques, which would utilize electronic signature during click-through acknowledgments in order to provide enough evidence during a possible dispute. When the advertiser receives a visitor coming from the publisher, it initiates a call back call to the publisher, which contains an ordered list containing the IP address of the user, the time and date of the referral, the page to which the referral was made, and the referring page – all signed with a digital signature. For performance reasons, they offer a modification of this techniques that uses hash chaining.

Although the problem of hit shaving and its possible solutions are interesting, hit shaving is not typically of concern in pay-per-click advertising as both sides have relatively reliable means of measuring the number of outgoing/incoming clicks and their source. Hit shaving would become an issue if the proportion of users with blocked referrers would increase or if a different model than pay-per-click was in place – such as pay-per-action.

A pay-per-action system is more susceptible to hit-shaving as the publisher looses sight of the visitor as soon as the visitor leaves the publishers website. The publisher has little means of knowing if for a particular visitor which traversed from P to A the desired action did or did not occur on A. Advertiser can then easily hide from publisher part of the conversions, which will result into direct monetary saving for A as the publisher P is, however, paid per action.

Hit shaving is likely to occur also unintentionally. For example, if a visitor comes from P to A and then later returns to A to buy the product, the commission will not be credited to P if the visitor cleared her cookies in the meantime. This can happen quite often according to a recent study [27], which observed "that 31 percent of U.S. Internet users cleared their first-party cookies during the month."

This weakness of the PPA model should be out weight by much more difficult fraud possibilities for a dishonest publisher. Closer inspection of hit shaving attacks is out of the scope of this paper. In the next section, we will show that although there are almost no

incentives for click-fraud in the PPA model, new kind of fraud – conversion fraud emerges.

1.2 Pay-Per-Action Model

Click fraud is often seen as one of the main drawbacks of the cost-per-click model. The following statement of the president of web consulting firm Alchemist media is representative for many other companies[2]:

> I don't see any point at which this issue will ever be resolved for either side with a total victory,... It'll be a constant battle as long as the current PPC [cost-per-click] model is maintained and isn't changed in some significant way.

The successor of the now very popular PPC model is widely recognized to be cost-per-action (CPA) model, sometimes also referred to as cost-per-lead or cost-per-conversion. This model is widely hoped to eliminate click-fraud as the publishers will be paid per conversion not per click. As we will see, the ability of the CPA model to meet these hopes largely depends on the definition of the *action*. Moreover, CPA shifts the risk from the advertisers to the publishers. Although the possibilities of the hit inflation attack might diminish, CPA might open a window of opportunity for the hit shaving attacks. In contrast to PPC, where the notion of click is common to all websites, a common definition of conversion/action is problematic.

1.2.1 Defining Actions

Before evaluating the resistance of pay-per-action model, it is needed to define the *action*. In ideal case, action would be defined as *hard* conversion – the visitor puts a product into a shopping cart, proceeds to a check out and pays for the goods with a credit card. The visitor is considered converted (the action happens) only after the transaction is validated.

This definition of action rules out almost any possible fraud[3]. Unfortunately, it is not always possible and not always desirable. First, it is applicable only for e-commerce sites. Second, the generally "hard" conversions happen quite infrequently. According to FireClick survey, the current[4] average conversion rate across all industries is only about 2%. For publishers with smaller traffic it would be unacceptable to wait for any monetary reward until a conversion occurs.

To illustrate this, take a median click-through rate on Paid search, which is reportedly 3.4 % for Jan. – Aug. 2006. Assuming that the CTR of the ads on the publisher's site also equals 3.4 % and the advertiser's conversion rate also equals the cross industry mean of 2%, the publisher has to generate approximatelly $1/(0.02 * 0.034) = 1470$ impressions to generate one conversion on a B2C e-commerce site.

In addition, there is always the possibility that the conversion occurs, but it won't be rightfully credited to the publisher. The visitor might make the purchase from a different computer or by phone. In that case, the advertiser will be likely unable to credit the purchase to the publisher, because she will not simply have the information that the visitor first visited the advertiser website by clicking on an ad on the publisher's website.

[2] According to `http://www.itmanagement.earthweb.com/columns/executive_tech/article.php/3410931`

[3] Save perhaps few relatively unlikely scenarios. For example, the visitor would pay for the product and after some time request her money back under some pretext. In the meantime, the publisher would receive the commission from the advertiser and would be reluctant to return it.

[4] Source: `http://www.fireclick.com/` Data for 31 April – May 6 2007.

1.2.2 Softening Actions

As we have shown, there is a need for *softer* definition of action. By using softer conversions, the advertisers provide incentives to publishers with smaller traffic. Soft conversion definitions are generally suitable both for e-commerce sites as well as for non e-commerce sites. Because a pay-per-action model which allows soft conversions differs from the conversion fraud perspective significantly from the PPA model allowing only hard conversions, we will mark the PPA model allowing also for at least one soft conversion as PPA*. Examples of soft conversion follow.

Requesting specific piece of further information In some cases, a visitor might be considered converted when she requests further information from the website. For example, the visitor fills in her personal data into a form and requests a hard copy of a product catalog. A strong interest of the visitor in buying the product can be inferred from the fact that she spent her valuable time to fill in the request form.

Registering for a newsletter The visitor may not find a suitable product, but may be interested enough to sign up for a newsletter offering promotions and new products.

Signing up A visitor is converted when she fills in a registration form and her email address is successfully validated.

Visiting a specific web page or set of pages for a specified minimum amount of time This may be a product detail page for and e-commerce site or a page with bank connection for a website of a charity.

As we have seen, the foreseeable definitions of actions are quite diverse. As we will show later, the softer the action criterion, the closer the CPA model is to PPC what concerns the susceptibility to attacks.

Further, we will assume that the publisher knows for which visitor actions she will get paid for by the advertiser. As we will see further, there could be a motivation for the advertiser to hide the definitions of the softer actions from the publisher.

1.2.3 Tracking Actions

Before we delve into possible attacks which might be carried out against the PPA model, lets first consider how the publisher can verify, whether a particular visitor committed any of the actions. For the simplest actions, such as visiting a specific web page, this information can be obtained from a log file. Generally, the action is a user interaction with a specific page, which may or may not result into a record in a server log-file. An example of an interaction which does not leave any trace on the server is when the user expands a part of a page, which is preloaded but would remain hidden if it were not for the user interaction.

Javascript client-side scripting language provides a common way to track user interactions. Javascript is also often used for tracking user actions in Web Usage Mining software[5]. When the tracking script intercepts a user action, it has to somehow tell this to the server. The most obvious way is by invoking a server side script by a HTTP request.

[5]Web Variable Instrumenter Program, `http://zing.ncsl.nist.gov/WebTools/WebVIP/overview.html`

It should be noted that Javascript generally fails to track actions, which happen in Macromedia Flash and Java applets included in some web pages. A concrete example of using Javascript is disclosed later in Chapter 5 in relation to describing how Google Analytics work. Google Analytics is currently one of the world's leading click-stream analysis solutions with some features of affiliate tracking software.

The action tracking function could be carried out by an affiliate system or clickstream analysis software. Action tracking is an integral function of most affiliate systems, while (arguably) much smaller percentage of clickstream analysis software implement it. We will further call software which tracks visitor actions in relation to the referral as "affiliate system".

1.3 Conversion Fraud

The two ways PPA could be compromised are analogous to hit shaving attack and hit inflation attack well known from PPC. For PPA, we could dub them *conversion shaving attack* and *conversion inflation attack*. Further we will only deal with the conversion inflation attack.

The character of the attack largely depends on how a particular advertiser defines actions (conversions) on her website. As we have already explained, perpetrating attack against PPA scheme requiring hard conversions, where a visitor is considered converted only after she has purchased and paid for a product, would be cumbersome for the attackers. Unfortunately, the hard definition of conversion is not possible or reasonable in many cases. Further, we will focus only on PPA schemes with one of the softer types of conversion.

The conversion inflation attack could be carried out either by humans, or by software.

1.3.1 Human Perpetrated Conversion Attacks

Despite the fact that there was limited rationale to employ relatively expensive human labour for attacking PPC advertising networks, when at least since 1999 there was a sophisticated software for carrying out click-fraud on the market [26], there were still many companies hiring people for this kind of "work" according to a 2004 newspaper article [83].

With the advent of CPA there might a temporary resurgence of this phenomena, as meeting the action requirements might in some cases require filling in web forms and overcoming captchas. This would require OCR readers and expensive wrappers; human labour might prove cheaper once again.

The advantage of attacks carried out by humans is that a human being can easily fulfill any action required by the advertiser's affiliate system. These attacks may be arguably more difficult to detect[6] and may not necessarily be too expensive. In a fraud free world where both the advertiser and the publisher are in hold of perfect information and the conversion rate is 2%, the price-per-conversion should be 50 times higher than the price-per-click.

The disadvantage of the attack perpetrated by humans is that the attackers would have to have available large number of undiscredited IP addresses. If the attackers used IP addresses from only a small pool, the above average number of conversion from these IP addresses might alert a fraud detection mechanism. Moreover, arguably the patterns in the behaviour of a small number of individuals paid per fake conversion might be more obvious to fraud detection mechanisms than patterns of well-written fraudulent software.

[6] Immorlica et al [48] claims that human lead attacks in PPC systems are "virtually impossible" to detect with a machine learning algorithm."

The attacks might also be easier to detect, when the attacker uses public proxy servers to perpetrate the attack. Although the conversions could be then distributed among many IP addresses, these IP addresses might have been discredited prior to the attack due to the fact that another attacker used them in a previous attack, which was caught by fraud detection mechanisms.

The software perpetrated conversion attacks discussed next, exploit the browsers of honest Internet visitors and thus they remove the problem with untrustworthy IP addresses. These techniques have, however, their own weaknesses and limitations.

1.3.2 Software Perpetrated Conversion Attacks

Fraudulent software could use a modification of the "undetectable" hit inflation attack proposed by Anupam [12] to perpetrate conversion fraud. In the original attack, a browser of a honest visitor of website S is secretly made to request a page from the dishonest publisher's website P, this page again secretly instructs the visitor's browser to request a page from the advertiser's website A. Because the request of V has P in its HTTP Referral field, A believes that this request is a click from P to A. P is payed for the click.

In the description Anupam gives about his attack, it can be derived that the attack is principally able only of simulating visits of length 1. Visit length is a number of pages visitor views during a session. If the required action can be fulfilled by viewing one page (e.g. viewing a page with bank connection) then this attack can be used in virtually unaltered form.

An action might however require viewing more than one page. In the Anupam attack, the browser of one visitor of S might be coerced to make multiple requests for several pages on site A. If the affiliate system on A tracked sessions properly, it would consider each of this requests as a new visit of length one. For example, the action requires viewing pages P1 and P1. If P1 is viewed during the first visit and P2 during the second visit of the same visitor, the affiliate system may not recognize the action. In addition, the traffic coming from site P would show some anomalies including but not limited to the following: the distribution of the visit length would be skewed toward short visits and the number of visits per visitor would be skewed toward larger values.

In some scenarios, carrying out a Javascript tracked user action (such as expanding a part of page with more detailed information) might be required. Although it is possible for sites S and P to load a page from A to the visitor's browser as a layer, frame or new window and thus simulate a pageview, these sites can not simulate user actions of site A. Anupam suggests that tracking user actions such as mouse movements is the most effective defense against the hit inflation attack.

Based on the information provided so far, the CPA scheme might seem more robust against computer-lead hit inflation attacks especially when the definition of the action requires a Javascript tracked user action not mere page view. As it show in the Chapter 5, the way the Javascript action tracking scripts communicate with the server might become a critical vulnerability.

The attacks against PPA model can be more feasible, if they are carried out in a manner tailored for the specific action tracking system in place. Therefore, it may be viable to use two independent systems to collect data, both using the same methodology to allow comparison of their results. A description of such system dubbed GAIN proposed and implemented by the author of this thesis is described in Chapter 6. Among other features, GAIN collects data on a very fine granularity. The data can then be sifted by a Web Usage Mining algorithm to

find suspicious conversions or clicks.

1.4 Conclusion

Conversion fraud can be feasible for the fraudster also when practiced only on a small scale when softer conversion definitions (such as requesting further info) are in place. As it was shown, the amount earned for one spoofed conversion should be in the order of a magnitude higher in the PPA* model than for one spoofed click in the CPC model. This might make conversion fraud significantly more difficult to detect with statistical methods, or machine learning/data mining algorithms, because these usually require large sets of data. While in CPC the fraudster might need to fraud 1000 clicks per 0.1$ in order to make additional 100$, in PPA* she would have to fraud only 20 conversions 5$ each to earn the same amount.

The inherent reason for the susceptibility of both CPC and PPA* model to attacks is the intangibility of the condition upon which the commission for the click or conversion is paid. Leaving only the hard conversions in place is not feasible for many websites as well. The best solution might be using PPA when possible and complement it with CPC model. Immorlica et al [48] has shown that a CPC model can be modified in a way removing incentives to fraud.

For the time being, applying data mining algorithms on web usage data provides one of the most powerful ways of detecting click fraud. Alexander Tuzhilin, a data mining expert who was invited to audit the measures Google takes to fight click fraud, recommended Google to adopt data mining algorithms on wider scale to be more effective in their battle against click fraud [81]. The insufficient amount of attention paid by Google to resistance to fraud of its products is documented in Chapter 5, where a proof of concept conversion fraud attack against Google Analytics is documented.

Data mining algorithms should be capable of detecting the undetectable hit inflation attack [12]. However, as was shown in Section 1.1.2 some approaches have unrealistic demands on the availability of data from ISPs. Chapter 6 presents a more realistic approach to collecting data in suitable granularity for click-fraud and conversion-fraud detection dubbed GAIN. This solution, proposed and implemented by the thesis author, is principally able to perform the most effective defense against the hit inflation attack [12], which is the detection of mouse movements. GAIN can provide data mining algorithms with data fine enough to allow them to spot the undetectable attack on the basis of detection of anomalies exhibited by the inflated hits; these anomalies are described in Section 5.2.

Chapter 2

Data Acquisition

This Chapter roughly corresponds to the CRISP-DM *Data Understanding* phase. Acquisition of quality semantically enriched data for Web Usage Mining is a challenging problem that requires applying knowledge from web-engineering and ontology-engineering. Data acquisition for clickstream analysis is often taken as a synonym for processing web server logs. It should be noted that this Chapter gives minimum attention to this topic; it shows a range of other approaches to collecting web usage data (and semantics) and compares them. Some of these approaches are commonly used in practice, some are only experimental.

The basic task in the data acquisition (or in some sources data collection) phase is to record visitor actions such as page views and on page events and the semantic information pertaining to it. Additional information may be gathered in order to identify user sessions[1] and assign weights to the actions.

The output[2] of the data collection phase is a set of n pageviews, $P = p_1, p_2, \ldots, p_n$ and a set of m user transactions (sessions), $T = t_1, t_2, \ldots, t_m$, where each t_i in T is a subset of P.

This chapter is organized as follows. It comprises of two sections, the first devoted to primary data and the second to metadata. Section 2.1 deals with the acquisition of the visitor actions and clickstreams, which is a well-described process in the literature. The character of Section 2.2 is less concise, because the acquisition of semantics for Web Usage Mining (WUM) is a new and quickly developing area, which has been ignored by mainstream publications on web metrics and web traffic analysis such as [34, 73].

2.1 Primary Data

The quality of the clickstream data available is an overlooked, but critical factor determining the success of any further processing. Data can be collected on the client side, on the server side, or both. Client side monitoring is required to get a precise record about a user interaction with the website [61].

Web server log-files are the most commonly known type of server side monitoring. This was a primary approach to clickstream analysis in the past. It can still provide a useful

[1] Identification of user sessions is usually (in-log file based approaches) a matter of the data preprocessing phase rather than data-collection, because no direct session identification data is usually available. Because this chapter focuses mainly on cookie-based approaches where session identification is rather straightforward and usually happens in real-time as an inherent part of the data collection, it was moved to this chapter.

[2] The mathematical notion was adapted from [59]

insight, although it is becoming obsolete. Server-side monitoring is not, however, a synonym for web server log-files. A server-side application may log any interaction (or its implications) with the visitor which is propagated to the server.

2.1.1 Comparison Of Client-Side Tracking And Web Logs

First, lets consider web logs, the doyen of web analytics, in comparison to a generic[3] client-side tracking. The focus is on the reasons why web logs are increasingly deemed as insufficient source of data and on the indispensable role web logs still play in accurate visitor tracking. Then we deal with some interesting visitor tracking solutions, which are not yet commonly used in practice (they are not highlighted in [73, 34]), but were published as research papers. Finally, there is an attempt to unveil the future threats to the client-side tracking as the apparent current market leader.

Before listing the disadvantages of web logs as compared to cookie-based[4] client side tracking, we make one strong assumption - web logs do not contain session identifiers. With this assumption in mind, we can list the principal disadvantages of web logs as compared to cookie-based client-side tracking:

1. The data are not readily available for analysis.
2. Inefficiently utilize disk space.
3. There is a number of heuristics for session identification [30].
4. The granularity of user interaction with a website is limited to a page level.
5. Some hits are omitted due to proxying and caching[5].
6. Unreliable tracking of visitors across multiple sessions.

Although client-side tracking is deemed more convenient and can provide far greater granularity than log-files, the server side tracking has not yet completely lost its place. The reason is the questionable accuracy[6] of client-side tracking. Although client-side can effectively prevent the omission of hits due to proxying and caching, it may typically omit or incorrectly sessionize hits from browsers exhibiting one or more of the following characteristics:

1. The browser has no support for cookies, or its cookie support is switched off or restricted to first-party cookies only.
2. The browser does not download images, or its image downloading function is turned off.
3. The browser does not support javascript or it javascript support is turned off.
4. The browser is – usually by means of some plug in – configured to purposely prevent execution of a specific client-side visitor tracking solution.

[3] Arbitrarily represented Google Analytics `http://www.google.com/analytics`

[4] A cookie is a piece of information sent by a Web server to a Web browser. This information is stored by the browser on the client's computer and may contain various information specific to the web application including the session identifier.

[5] Occurring frequently when the user hits the browser's back button.

[6] The accuracy is here defined as the proportion between the real number of hits and the observed number of hits.

As contrasted to server-side tracking, which is difficult for the visitor to avoid, even a visitor with little or no technical skill can – both intentionally and unintentionally – avoid being tracked by a client-side solution most frequently by disabling cookie support in her browser. However, many modern e-commerce web sites are build with the expectation of the visitor having support for cookies, images and javascript and do not work well when the support for any of these is missing. Out of the reasons listed above, only no. 1 is currently considered as an issue [4] in client-side tracking. The danger to the accuracy of web analytics posed by client-side tracking blockers is discussed in further in the text.

Web-logs can complement client-side tracking because the former is largely inert to a client-side support for cookies and javascript. Some tracking solutions, such as Urchin Tracker combine both log-file and client-side tracking to ensure maximum accuracy. Google Analytics which is extensively referred to further in the text, especially in its case study part is built upon the Urchin solution. However, it does not use web-logs[7].

It should be noted that there is a work-around, which allows exact session identification even when cookies are disabled. URL rewritting can be used to label every visitor's request with a unique visit-specific ID by including it in the URLs.

2.1.2 Client-Side Javascript Tracking

The most well known form of client-side tracking uses javascript tracking code, which is a part of the website. Tracking is executed on the client side, but usually the process is absolutely transparent for the user, who does not know that she is tracked and, particularly, does not need to install any tracking software.

As an alternative to javascript tracking, some researchers (e.g. [70]) use a Java applet, which provides them with a less constrained environment while maintaining the key advantage of javascript tracking – no need for client-side installation. However, for Java applet based tracking speed as well as client Java support becomes a concern.

Some research projects (WebVIP[8] or WET [32] extend the usual javascript and tracking icon approach[9] to gather more information about user actions than is usual in commercial packages.

Both WebVIP and WET were developed at the U.S. National Institute for Standards and Technology (NIST) within its WebMetrics project[10] of the Visualization and Usability Group. Based on the information provided on the website, this group has not been active since 2003. The outcome of their research, including WebVIP and WET posed a remarkable achievement at the time. In addition, it is interesting to explore the reasons that hindered a wider adoption of these free tools. Next, a brief review of the WebVIP and WET systems is given.

[7] This solution apparently collects some data (e.g. User Agent) by a server side approach, but only as a part of execution of the client-side tracking module. As a consequence, there is no increase of accuracy. The predecessor of Google Analytics, Urchin Tracker offered also the option of combining information from web logs and client-side tracking.

[8] http://zing.ncsl.nist.gov/WebTools/WebVIP/overview.html

[9] A reference implementation which is in detail described in [53].

[10] http://zing.ncsl.nist.gov/WebTools/index.html

WebVIP

Web Variable Instrumenter Program (WebVIP) is intended primarily for usability engineers. It supports logging of on-page events, such as page loads, mouse overs or clicks. This software can be used exclusively on static websites comprising of a set of physical HTML pages. It copies a website to a test directory, parses the HTML pages and adds JavaScript code (custom event handlers) to the pages. The usability engineer can choose, which events will be logged. When the visitor interacts with the test copy of the website, the selected events are logged and stored into sessionized FLUD log files.

Framework for Logging Usability Data (FLUD) is a file format devised also within the NIST Web Metrics testbed. Its complex structure allows it to attach multiple actions to one page view. Apart from storing event-handler related info, it also support storing information pertaining to higher-level user-page interactions, such as filling in a questionnaire.

The authors acknowledge that the Web VIP system has some disadvantages[11]:

- Although event handling code is changed automatically for each of the pages, the process still needs to be overseen by a usability engineer, because of possible conflicts with existing javascript event handlers.
- Copying the whole website for test purposes is cumbersome.
- Forms and elements other than HTML links are not supported.

Getting WET

Web Event logging Tool (WET) was an attempt of the NIST Web Metrics TestBed to overcome the drawbacks of Web VIP system. It can be said that WET has already most of the features of a modern web tracking solution. Its installation is easy requiring only uploading a javascript file to the server and including a reference to it on each of the tracked pages.

Next, the specific WET implementation needs to be tailored in order to capture only certain aspects of the user's interaction, like clicks on objects, changes to objects, mouseovers on navigation items, and page loads. Both static and dynamic pages are supported.

The WET system is intended for collecting data, it does define the way of storing and retrieving the data generated by WET, a range of server side technologies could be used to capture and persists the data sent by WET.

2.1.3 Server Side Monitoring

Web-logs are not the only form of server-side tracking. Some approaches use a application-level server side tracking meaning that the web server application responsible for the generation of web pages is also in charge of tracking.

For example, when the web server application generates the content of a dynamic page, it can record which pieces of information it contains. The representation of the page content is then not limited to mere URL as in the web log based approaches. An authoritative survey [33] marks application-level server side tracking as "probably the best approach for tracking Web usage"'.

Custom server side monitoring has several important virtues, when we consider its deployment in a dynamically built website:

[11] http://zing.ncsl.nist.gov/hfweb/proceedings/etgen-cantor/index.html

1. Transparency: no demands on the client, including scripting support.
2. Invisibility: the client can't know (in principle) whether she is tracked and what data are collected.
3. Speed: the data are readily available on the server-side, no additional client-server communication needed.
4. Interlinked with dynamic pages: all information in the underlying database, from which the pages are generated, is available for tracking purposes.

Despite these advantages, custom server side tracking has apparently never gain much popularity in e-commerce applications. It suffers from the same accuracy problems (what concerns the omission of hits) as web log based solutions. The other reasons are that it is costly to setup as each WUM application must be tailored for the specific server, it requires periodic maintenance as the scheme of the underlying database and the server application itself undergoes frequent changes. In some cases, the utilization of application-level server side tracking is further aggravated due to the copyright of server applications [33].

Another common form of visitor tracking is collecting data from proxies or putting a tracking code into the proxy tier. This solution is extensively used in practice by large companies. E.g. Google uses Google Web Accelerator which acts like a proxy. The Google Web Accelerator Privacy Notice says that: "'..we may use log information about Google Web Accelerator usage to improve the quality of Google Web Accelerator and other Google services.". Although the morality of this is questionable, the user usually gives the consent to being tracked in the terms and conditions she signs with her service provider.

For the completeness sake, there is also the possibility to track visitors with TCP/IP packet sniffers. This approach is too low-level to be used in practice [33].

2.1.4 Alternative Approaches To Client-Side Tracking

There were numerous approaches published in scientific papers studying user interaction with websites. Generally, it can be said that the more granular the data the more demanding the experiment set-up. Scientific research in WUM paves the way for commercial applications, but in most cases they can not be directly reimplemented in an e-business setting, because of their high knowledge acquisition costs, difficult interpretation, and last but not least the little monetary benefit the outcome of such a research would bring to the business.

The scientific researches presented here are hence not meant mostly for reimplementation in a business setting, but rather for their interesting conclusions. The researches are sorted by the method they use or rely on for data collection, however all their phases are presented in this chapter (although they should logically belong to the following chapters) for the sake of better understandability.

Client-Side Tracking

The top-of-the-line approaches need a special hardware to perform eye-tracking in order to locate the exact part of the page the visitor looks at. A relevant research to e-commerce WUM using eye tracking is presented in [42]. This paper studied the relation between a rank of the link and the attention it receives. Surprisingly, the authors found out that the time is almost equal for links number 1 and 2. This is contrast to the fact that numerous researches

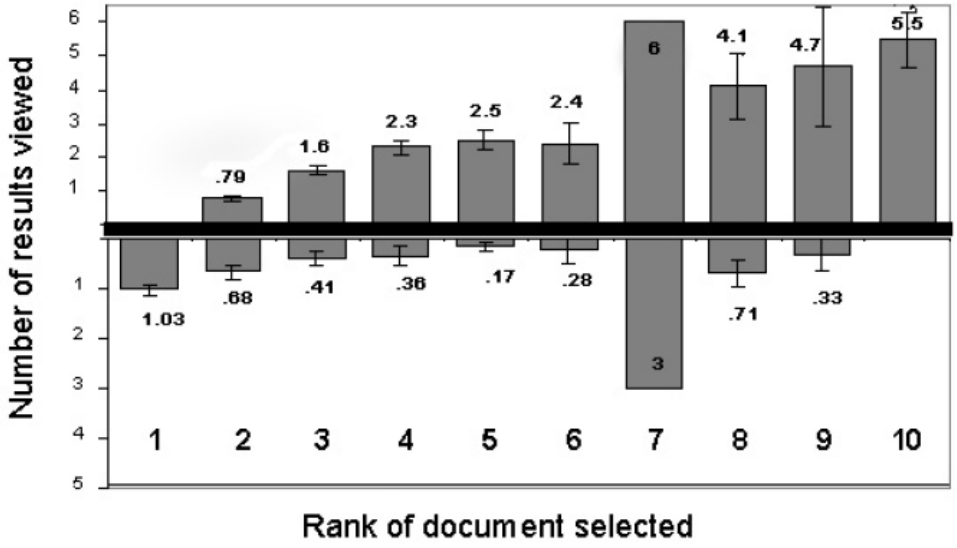


Figure 2.1: The number of results above and below the selected document which users scan on average. Source: [42]

and statistics (e.g. statistics of ad schemes `sklik.cz` and `etarget.cz` show that users click more often on the first link rather than on the second link. The experiment also confirmed the widely held assumption that visitors scan results from top to bottom.

Other approaches utilize a standalone software or a browser plug-in to track user interaction with a website. According to [16] this is the case with research presented in [80].Client-side tracking software is often used for usage pattern analysis, because it gives a possibility to gather very detailed information, which would not otherwise be accessible. This information is used mainly for the following types of tasks (according to [49]):

- Recognition of Common Areas in a Web page Using Visual Information
- Vision-based Page Segmentation

A very interesting research relevant to WUM is presented in [49] which builds both upon Recognition of Common Areas and Vision-Based Page Segmentation. Unlike some of the previously mentioned research methods which are dependent on arbitrary client-side software, this approach apparently puts little demands on the client setup, because it probably utilizes javascript.

Recognition Of Common Areas in A Web page Using Visual Information

Milos Kovacevic [55] uses a set of arbitrarily defined heuristics to locate page *Common Areas* – page Header, Footer, Left Menu, Right Menu and Center. This approach is based on a visual analysis of the page as seen by the visitor in a browser. The goal is to label each HTML object as belonging to one of the H, T, LM, RM or C classes and then use this classification for weighting the text contained in the element.

The researchers define the M-Tree format of a page used to render the page on a virtual screen obtaining coordinates for every HTML object. An example of an object on a finest

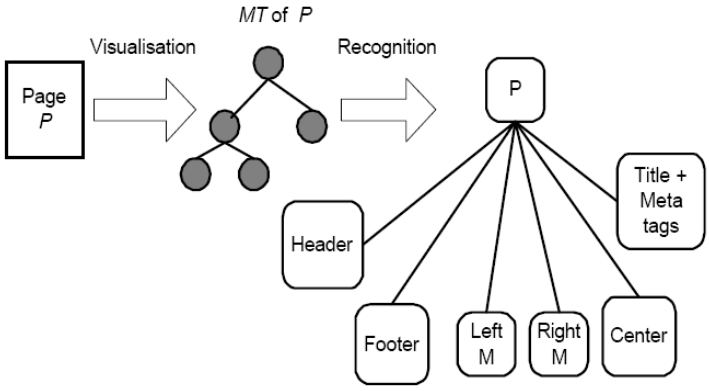


Figure 2.2: Page representation using visual information (Source: [55])

granularity level is a table cell. A rule-based heuristic is used for identifying elements belonging to each of the set of Common Areas. The application of each of the H,F,LM,RM and C heuristics is constrained by a visual interval (in pixels) within which the individual Common Areas may be exclusively found. These intervals were found by a statistical analysis of a sample of web pages.

The usefulness of labeling each of the page HTML elements by Common Area and consequently using these for weighting purposes was tested on a page topic classification. When using a Naive Bayes classifier, the information carried by the visual appearance increased the classification accuracy by more than 10%.

This method has several advantages over other approaches to weighting HTML page elements, especially to those that are based on information retrieval techniques over a DOM tree. Many HTML pages do not comply with X/HTML standards, and even if they do, the semantics expressed by HTML tags and the DOM tree is limited[12], because HTML was designed mainly for page rendering purposes. On the other hand, the authors claim that their method is more reliable – the information value of the Center part of page is significantly higher than of the remaining parts. Similarly, links in the footer of the page are less likely to be semantically related to content of the page than links in the center.

Page visualization may also add some robustness to existing information retrieval techniques, because it may help to detect keyword stuffing – invisible text, which is often a sign of deliberate attempt to manipulate the page ranking on a search engine result page.

Vision-Based Page Segmentation

The idea behind vision based page segmentation is to mimic the way how humans visually process web pages. The basic assumption of this approach is that when a web page is presented to the user, she uses spatial and visual cues to unconsciously divide the web page into several semantic parts. The VIPS system [23] automatically segments the web pages by using the spatial and visual cues. VIPS searches for separators in order to split the page into segments. Separators are horizontal or vertical lines in a web page which visually cross with “no blocks”. The notion of Degree of Coherence (Doc) is defined in order to measure how coherent is the

[12]This statement is underpinned by an experiment in [23].

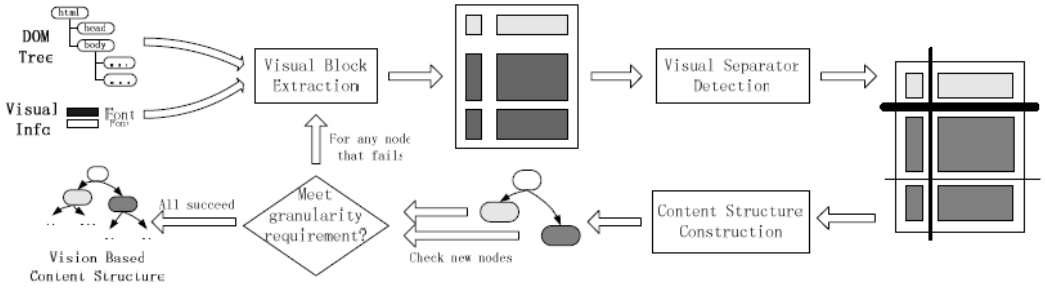


Figure 2.3: Vision-based Page Segmentation Algorithm (VIPS) [23]

content of the block based on visual perception. A threshold value for DoC is used to control the granularity of the segmentation. The DoC values are assigned to each extracted block based on the block's "visual property". In the block extraction phase, heuristic rules are used to judge whether a node should be divided. These heuristics are based on tag, color,text and size cues. For example the text cue means that if most of the children of a DOM node are text nodes or virtual text node, it is preferred not divide it. An example heuristic is:

> If all of the child nodes of the DOM node are text nodes or virtual text nodes, do not divide the node.
>
> If the font size and font weight of all these child nodes are same, set the DoC of the extracted block to 10.

2. Otherwise, set the DoC of this extracted block to 9.

The authors claim that the top-down approach of the algorithm is very effective. The experiments were carried out on query expansion domain in the field of information retrieval. The VIPS algorithm can group semantically related content into a block. The experiments confirmed the baseline assumption that the term correlations within a segment will be much higher than those in other parts of a web page. In addition, the results indicate that the average retrieval precision can be improved after partitioning pages into blocks, no matter which segmentation algorithm is used.

Tracking Exposure Time And Clickthroughs

Having explained the two underlying concepts, we can finally proceed to the papers [49, 37] named "A Study on Web-Usage Mining Control System of using Page Scroll" and "Design and Implementation of Web Usage Mining System Using Page Scroll" respectively. It should be noted that these two papers differ only in the name, their content (on the wording level) is except for one additional image in the latter identical.

This system probably uses javascript for client side tracking. The use of javascript is, however, not explicitly mentioned in the article, but it can be inferred from the fact that the tracking part of the application dubbed User Activity Collector (UAC) is downloaded by the client. Also the exchange format the UAC uses to transmit the data to Vision Based Usage Mining server (VUM) is characteristic for javascript/AJAX applications.

```
url=/Interest/49.html,referrer=http://www.eyerank.com/
```

```
Interest/Interest.htm,time=2005-6-15/13-57-.858,zone=-9,sw=1280
,sh=1024,bw=1272,bh=915,color=16,cookie=y, java=y,history=0
```

The result of the analysis is to determine which area of the page is most exposed to the user. This information might be very valuable to marketers, for example for determining the price of advertisements depending on their position on the page.

The Disadvantages Of Client-Side Based Tracking Software

The analysis of data collected by client-side based tracking software approaches can provide highly interesting results, but these approaches exhibit high cost per tracked user, additionally the number of users tracked usually small. It must be said that there were numerous attempts to collect data on the client side on a larger scale by private businesses. Many of these client-side tracking solutions operate without user consent (or understanding) and consequently classify as spyware, sometimes even trojan horses or plainly viruses. Probably the most successful solution of this type is maintained by Alexa.

The service provided by Alexa is generally provided on and opt-in basis. The tracking module is embedded into a browser toolbar, which provides useful functions to the users with the tradeoff that it collects data during their browser usage. Alexa argues[13] that their service is legal and cannot be classified as spyware, because the user gives consent to being tracked during the software install.

This does not, however, prevent a lively discussion on the Internet[14] whether Alexa toolbar is a spyware or not. It can be expected that with growing privacy concerns, this kind of data collection will be seen as increasingly intrusive and will be, as such, a rewarding target of many security tools.

2.1.5 Blocking Client-Side Tracking

The significance of "client-side tracking-blockers" is, according to the opinion of the author of this thesis, underestimated. Although not currently being an issue, major browsers allow the installation of privacy protecting browser plug-ins, which prevent the execution of major client-side tracking code. The question is – will these "client-side tracking-blockers" become successful? By a "client-side tracking-blockers" we understand a browser feature that prevents or hinders client side tracking while not significantly limiting users Internet experience.

Lets consider several parallels from the history.

In recent years, we have seen a massive adoption of pop-up killers. These eventually became a feature of major browsers. Pop-up windows are a nuisance to the visitor, and some even indirectly provide a security risk. Another example are ad-blockers, which automatically remove banners from websites. The current adoption rate of ad-blockers is nearly negligible, because neither Internet Explorer, Firefox or Opera have an ad-blocker built-in. The reason why ad-blockers did not repeat the success of pop-up-killers is, according to the author of this thesis, two fold.

First, the users do not probably perceive ads as being as harmful and undesirable as pop-ups. Second, ads are an important source of revenue for content providers and ad-companies. Microsoft being a content provider and running its own ad-scheme has obviously little interest

[13] `http://www.alexa.com/site/help/?index=74`

[14] Google query for "Alexa spyware"' results into 2 130 000/25 500 hits (the latter is a phrase search).

in including an ad-killer into its free browser. Although it may seem that Mozilla foundation, the body behind the Firefox browser, may be indifferent to commercial pressure,it is not probably so. Its strong ties with Google [39], the provider of a major ad-scheme, would undermine any effort to include a built in ad-blocker into a Firefox release.

Considering the historical experience, will "client-side tracking-blockers" follow the successful example of pop-up killers or remain without wide adoption such as ad-blockers?

Current browser vendors are unlikely to support tracking-blockers, because client-side cookie tracking is widely used among advertisers to measure the impact of their campaigns. Loosing this feedback channel might lead to a drop of confidence in on-line advertisements and consequently a drop in revenue for many stakeholders across the on-line advertising industry.

However, as Internet literacy is growing, the users are quickly becoming concerned with their privacy on the Internet. In addition, an increasing number of users is able to change browser settings and install plug-ins. Perhaps the most popular one is a *CustomizeGoogle*[15] plug-in for Firefox which among other things cripples Google Analytics client tracking by disabling its cookies. This plug-in quickly spreading with more than 5 million downloads reached by August 2006 [1].

As a conclusion, I believe that in the near future we can expect that the adoption "client-side tracking-blockers" *will reach the threshold above which their existence can no longer be neglected by web analysts and the vendors of the tracking tools.*

2.1.6 Data Cleaning

Data cleaning is much more important for server side tracking approaches than for client side tracking. There two major data cleaning tasks – removal of robot traffic and removal of irrelevant logged events.

Client-side tracking usually transfers the data to the server in a query-string of 1x1 image. Most robots (except for image-bots) do not download images, therefore the vast majority of robot traffic is ignored from the very beginning. Log-file based approaches have to, however, deal with this problem. Robots are usually identified by remote hostname, user agent string or checking the access to the robots.txt file.

These methods usually fail when the robot deliberately masks itself as a common browser by sending a false user agent string in its HTTP requests. These cases need to be taken care for by applying a heuristics. Research [78] gives evidence that robots take breadth first navigation in the tree representing the Web site and that they do not assign any value to the HTTP Referrer field in HTTP request. This information may be used as a baseline during the construction of robot detection heuristics.

In log-file based approaches, additional entries of no interest (typically logged requests for images) are removed. In cookie based approaches, only desired actions are logged, so this step does not need to be carried out.

It can be concluded that although there are ways of reliably removing the robot traffic, these depend on a provision of additional information (current list of robot user agent strings) and heuristics. In contrast to log-files, cookie-based client-side tracking does not almost need any data cleaning. The traffic from the few image bots could be removed by constructing the absolute part of the URL of the tracking image in javascript.

It is currently highly unlikely that an image bot would have a javascript interpreter. As

[15] http://www.customizegoogle.com

a consequence, the traffic of image bots would also be ignored and not sent to the tracking server. The side effect would be the omission of the relatively small[16] percentage of human visitors with no or disabled javascript support in their browsers.

2.1.7 Choosing The Right Tracking Method

Out of all the approaches listed, the author of this thesis believes that extending the traditional javascript and tracking icon approach is the most reasonable approach for most practical e-commerce-related analysis tasks in the Small&Medium Enterprises sector. However, if this kind of tracking is chosen, it is necessary to continuously follow the development of the number of users with client-side tracking blocker turned on. Ideally, client-side tracking should be complemented by web-logs.

2.2 Meta Data For Clickstream Analysis

Clickstream collection described previously is mainly about trying to obtain the paths and actions the visitors take when navigating through a website. Researchers increasingly recognize the positive influence semantic descriptions of pages in the clickstream can have on the quality of the data-mining result.

A quote from the recent meta-study "Mining interesting knowledge from weblogs: a survey" by Facca and Lanzi [33] is an apt description of the potential of the semantic information in WUM:

> ... we believe that the most interesting research area deals with integration of semantics within Web site design so to improve the results of Web Usage Mining applications. ... Efforts in this direction are likely to be the most fruitful in the creation of much more effective Web Usage Mining and personalization systems that are consistent with the emergence and proliferation of the Semantic Web.

2.2.1 Meaning of Semantics On A Website

By semantics we mean here a structured description of the primary content, structure and user data. The *content* data is usually a combination html and pictures, while the *structure* is a designer's view of the content organization. The quality of the *user* data is arbitrary. It can range from mere user location and bandwidth derived from the user host (IP) address to a comprehensive visitor profile for web site's registered users. Such profile may contain user history, purchases, rating of various products as well as other implicit or explicit representation of user's interest.

In clickstream analysis, the data being described are clickstreams - the URIs of the pages in the user's navigational path and more recently also the actions the users take within one URI. The URI's represent some resource. For data-mining purposes, we are interested in a structured description of the content of the resource identified by the URI, its relation to other resources and optionally in interaction with the user profile. In fact, we are seeking an *ontology* and a mapping of the instances (resources) to it.

[16] According to a representative survey taken in March 2006 and March 2007 by `thecounter.com` the number of visitors with disabled javascript rose by 1% from 3% in 2006 to 4% in 2007. It is not clear if these numbers include robot traffic.

Ontology is a "data model that represents a set of concepts within a domain and the relationships between those concepts"' [85]. A similar definition is also e.g. in [69], which also notes that a relational schema for a database is an example of an ontology. Tables may express concepts with columns expressing properties. These properties might be either atomic or may point to another concept (table) via a foreign key effectively establishing a *relation* between concepts. The three most commonly used types of relationships are: *generalization/specialization*, *aggregation* and *binary relationships*.

The rows contained in the individual database tables may be the instances of the ontology expressed by the relational scheme. A mapping of an instance to concepts is an important notion. In our simplified relational database example, the the mapping is determined by the membership of a row (instance) to a table (concept).

Unfortunately, we don't usually deal directly with ontologies or relational databases on the web. The content of the resource is typically written in the HTML language, which is not structured and granular enough for WUM. There are several basic options how to deal with this problem. It is more likely to encounter their mix in real world situations.

The most favourable situation (in practice) is when the website is dynamically generated from a relational database[17], which is well-structured. In that case, the db query language can be used to obtain a structured description of the website pages. In other words, we put the database to another use, when we use it as a source of an *explicit domain ontology*[18]. The drawback is that if the database schema changes, its mapping to the semantic layer has to be updated as well.

If the website is static and the database is not there, or its structure is not meeting the requirements, obtaining semantics is not so straight-forward. An *implicit ontology* may be acquired from a website by applying machine-learning and web mining techniques. An alternative is to design the ontology manually, but this is economically viable only for small sites.

Ontology can help to represent the events of a website. The URL of resources or the names of the on-page actions (the *physical level*) can be associated a meaning and mapped to a concept in an ontology (the *conceptual level*). There are two types of ontologies in Web Usage Mining – one representing service dimensions the other representing the content dimension [82, 53], [30] has been the first to employ content based information to enrich web log data[33].

There are many variations of this basic notion, in [82] a resource might be mapped to one or both dimensions, but if mapped to both dimensions one dimension must be labeled as the master dimension for a given resource. Only a selected granularity level of the master dimension is used for mining the patterns. As noted earlier, the service-content approach can be somewhat mapped to the research from the pre-semantic web age presented in [30]. There are five main types of pages (Head, Content, Navigation, Lookup, Personal) each having its own set of characteristics. Perhaps the most general approach out of the three presented is in the [51], where both content and service can be used on several granularity levels in the data mining process.

Figures 2.4a illustrates the process of mapping the resources to an ontology (here by running a stored procedure called Segment further described in [52]) and 2.4b gives an example

[17] Less frequently also by a transformation from XML file(s).

[18] It may include the product hierarchy or the navigational structure of the site as captured in the relational scheme or even directly in an ontology language such as RDF.

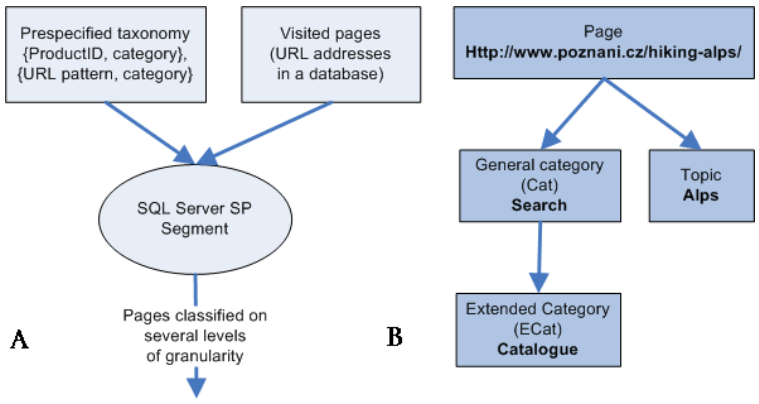


Figure 2.4: Mapping web resources to content and service ontologies

of how a particular page would be mapped according to the Service (left) and Content (right) *ontologies*. In the later text the term dimension will be used to refer to a concept on a given granularity level within the content and service ontologies. E.g. the content ontology may have "type of trip" and "country visited" dimensions on the lowest granularity level. The term *dimension* is used because of of the vector representation of the semantic information used throughout the remainder of this thesis.

2.2.2 Domain Ontology Acquisition And Knowledge Base Construction

Obtaining The Knowledge Manually

Many papers ([82, 51]) which are not focused directly on the data-acquisition rely on manually designed ontologies. These papers do not usually explain which tool was used for creating the ontology and mapping instances to it, how easy it is to maintain the ontology using this tool and how scalable this solution is. Usually, these ontologies are designed by a domain expert, web master and a knowledge engineer.

For most dynamically build web sites at least some of the information is readily available in machine-readable form in the underlying database or scripts. In a typical case, it might be necessary to build the ontology manually. It is highly desirable that the mapping between resources and ontology concepts be done at least partially automatically, considering the excessive amount of human labour which would be needed to map every resource (page, action) on a website to the content and service ontologies. The manual acquisition of knowledge should preferably mean programming application-specific semantic data adapters that will be able to provide the mapping information based on the already available information (such as database tables or classification tags included in the page source by the designer [30]).

It is definitely quite costly for most companies to create and maintain ontology for their web sites and the ontology mapping for each page. If a company has to exert the effort to semantically describe its data, its investment should be leveraged by using standardized formats (RDF and OWL). The key criterion is to make things as easy as possible for the user. This can be achieved by providing a mechanism for simple ontology creation, which will enable the web master to create a task specific ontology for the web site and link pages to this ontology. Currently, there are several tools, which facilitate ontology design and storage of the semantic data. The most widely known are the Protégé Ontology Editor and Knowledge

	Jacobs	Plus	classic	mild	light	< 6 DM	< 8 DM	> 8 DM
Dallmayr Prodomo			X					X
Jacobs Krönung	X		X				X	
Jacobs Krönung Light	X				X		X	
Jacobs Krönung Free	X				X		X	
Jacobs Krönung Mild	X			X			X	
Jacobs Meisterröstung	X		X				X	
Tempelmann			X					X
Plus Schonkaffee		X			X		X	
Plus Naturmild		X		X		X	X	
Plus milde Sorte		X		X		X	X	
Plus Gold		X	X			X	X	
Idee Kaffee Classic			X					X
Kaffee Hag klassisch			X					X
Melitta Cafe Auslese			X				X	
Melitta Cafe Auslese Mild				X			X	
Kaisers Kaffee Auslese Mild				X			X	

Figure 2.5: Formal context about coffee brands sold in a supermarket (source: [77])

Acquisition System[19] and Sesame – the RDF Schema-base Repository and Querying facility[20].

It is a matter of further discussions which specific approach would suit most the e-business user. In any case, the (expensive) work on the ontology should be preserved in a standardized way to allow its reuse for some other purpose such as semantic search in the future.

Obtaining The Knowledge (Semi-)Automatically

If there is no direct source of ontology, such as relational database scheme, automated methods of knowledge acquisition such as text-mining come in useful. The survey [59] lists the following techniques, which have been used for automatic generation of machine understandable ontologies for web sites:

1. Obtain concept hierarchies by applying hierarchical clustering algorithm to terms
2. Derive a concept lattice by applying the *Titanic* algorithm introduced in [77][21].
3. Learn generalized conceptual relations by applying association rule mining.

These techniques fall within the Web Content Mining area and are suitable for automatically creating a content ontology. Lets briefly develop only the notion of the concept lattice [41]. Concept lattice is a structured graph for representing knowledge, so that the structure can be used in the representation system. Concept lattice for given data (figure 2.5) can be drawn as Hasse diagram (figure 2.6).

[19] `http://protege.stanford.edu/`
[20] `http://www.openrdf.org/`
[21] According to [59] this algorithm is a variation of the association rule mining algorithm.

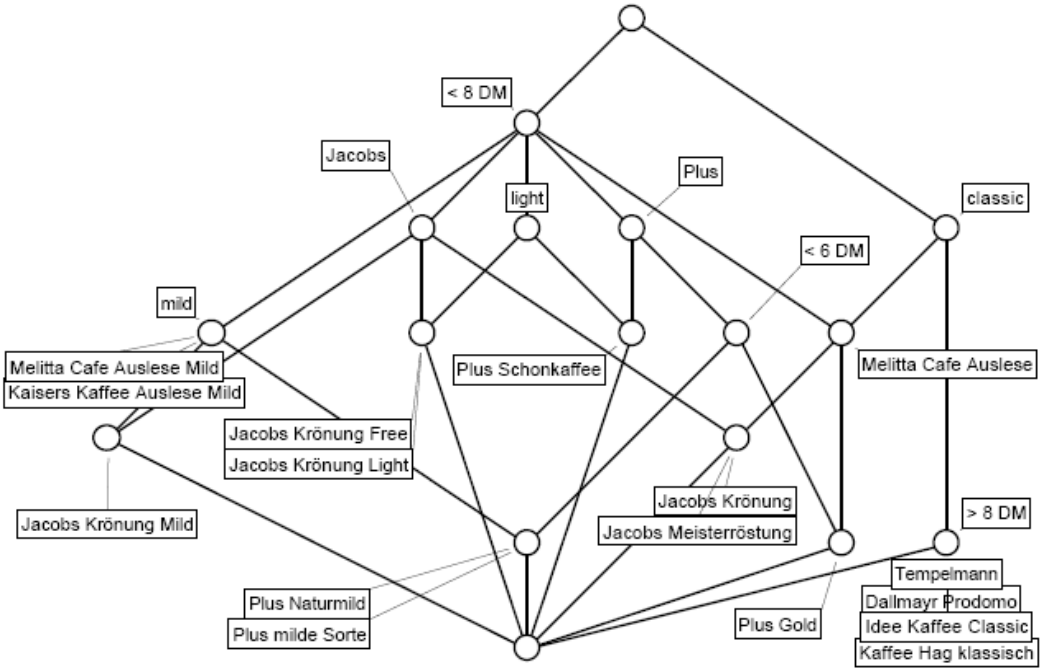


Figure 2.6: An example of concept lattice. Source: [77]

In the case study part of this thesis, there is an in-detail description of the LumberJack web analysis system, whose backbone is a multi-modal hierarchical clustering algorithm, which takes as one of the modalities a page term vector.

The service ontology could be obtained by applying Web Structure Mining techniques. Web Structure Mining deals with the structure of the hyperlinks within the Web itself [31], it can be used both for inter- and intra-site analysis.

The most-well known algorithms used in Web Structure Mining are the PageRank algorithm and HITS algorithm [31]. When applied an intra-site data, the HITS algorithm outputs an authority and hub value for the pages analyzed, based on the structure of links on the website. The authority and hub values are defined recursively. The authority value is computed as a sum of hub values pointing to the page. Similarly, the hub value is computed as a sum of authority values of pages it points to. The ranking of the pages produced by the HITS algorithm may help to classify pages on the service domain. Several groups may be defined based on combination of authority and hub values. These automatically groups may roughly correspond to the Head, Content, Navigation, Lookup and Personal page types mentioned earlier.

Ontology Representation Language

A choice of ontology representation language determines the machine understandability, the power of reasoning and the computational efficiency of the model. Although some approaches also use a relational model, [59] introduces a range of other approaches including vector-space model, first order logics and descriptive logics (DAML+OIL).

It must be emphasized that the ontology is only a conceptual description of the underlying domain which includes only the concepts it is comprised of and the relations between them.

It does not contain a mapping between objects on the web and the ontology. The process of creating this mapping is called *knowledge base construction*. A list of techniques (such as learning a text classifier) used for this task is available in [59].

2.2.3 Costs And Benefits Of Semantics

It is a widely acknowledged fact that background knowledge can help to improve the results [33], interpretation possibilities [82] and performance [15] of the data-mining process. Unfortunately, this does not come at no cost; knowledge acquisition is usually an expensive process. In addition, different pieces of knowledge have different prices and their positive effect also varies. There was some research done in order to identify some "rules of thumb" in order to facilitate the process of knowledge acquisition. The main resource for this subsection is the research [15]. First, a brief introduction to the different flavors of background knowledge is needed.

In the previous text we have narrowed the term *metadata* to semantics or ontological knowledge. However, other types of background knowledge can be very useful in the data-mining process. According to [15] there are in addition to the Ontological knowledge another two *classes* of the background knowledge – Constraint Knowledge and Abstraction Knowledge. Each of these three basic classes of knowledge can be further divided into several *types* [15]. Please note that some of these types were constructed and need to be constructed with a specific data mining method[22] in mind.

Constraint knowledge can be applied for example for filtering patterns by their quality and for restricting the search space. There are the following types: Language constraints, Quality constraints, Value exclusion constraints, attribute exclusion constraints, Value aggregation constraints, Attribute combination constraints, Priority groups and Subgroup patterns constraints.

For example, for association rules as the data-mining algorithm, the *language constraints* can restrict the maximal number of conjunctions in the antecedent or consequent of the rule, while the *quality constraints* set boundaries for statistical significance thresholds such as confidence and support.

Ontological knowledge was already described as "a conceptual description of the underlying domain". Research [15] approaches the ontology from a rather different angle than in the previous text when trying to extract pieces of knowledge that could basically help with setting the parameters of the data mining process. The types of ontological knowledge are: Attribute weights, Abnormality/Normality information, Similarity information, Ordinality information and Partition class information.

For example, the normality information expresses whether a specific attribute value is considered as normal (boolean attribute). Abnormality information is more detailed having a range of values for different degrees of abnormality. The ordinality information specifies "if the value domain of a nominal value can be ordered" [15][23].

[22] Subgroup discovery in the original paper [15]. Subgroup discovery is in some respects very similar to association rules, so the conclusion drawn and types used for the Subgroup discovery domain should (with some degree of cautiousness) hold also for a more commonly used algorithm in Web Usage Mining – for association rules.

[23] The author probably meant not only a boolean value as expressed in the definition, but also the order of the attribute values in case the nominal value can be ordered.

Knowledge		Derivable knowledge	Costs		Search	
Class	Type		Syn.	Cog.	Restr.	Foc.
CK	Language constraints	-	-	+	++	+
CK	*Quality constraints*	-	-	++	++	++
CK	Value exclusion c.	-	(+)	+	+	+
CK	Value aggregation .c	-	(+)	+(+)	++	+
CK	*Attribute exclusion c.*	-	(+)	+(+)	++	++
CK	Attribute combination c.	-	(+)	+(+)	++	++
CK	Priority group c.	-	+	++	-	+
CK	Subgroup pattern c.	-	+(+)	+(+)	-	++
OK	*Normality information*	Value exclusion	+	+	++	++
OK	Abnormality information	Value exclusion	++	++	++	++
OK		Value Aggregation			+	++
OK	Similarity information	Value Aggregation	+(+)	+(+)	++	++
OK	*Ordinality information*	Value Aggregation	+	+	+++	++
OK	Attribute weights	Value exclusion	(+)	+(+)	++	++
OK	Partition class i.	Value combination	+	+	++	++
OK	***Derived attributes***	Derived attributes	+++	+++	-	+++

Table 2.1: Applicability, benefits and costs of individual knowledge types

Abstraction knowledge is composed by derived (computed) attributes. According to [15] there are three main effects of the derived attributes:

- Focus the data mining method on the relevant objects for the analysis.
- Decrease multi-correlations between attributes that are not of interest.
- Can reduce the number of missing values, because derived attributes can be computed in such a way that even if one of the underlying values is missing its value is defined.

The classes and types of background knowledge (CK=Constraint Knowledge, OK = Ontological Knowledge, AK = Abstraction Knowledge) have different characteristics in terms of their intrinsic knowledge value, their syntactical and cognitive costs and their potential to restrict the search space or focus the search process. The following notation is introduced in [15] : sign "-" stands for no cost/impact, signs "+","++", "+++" stands for increasing costs and impact, "+(+)" indicates that the element is cheap if it can be derived or learned and moderate cost otherwise.

According to empirical experience presented in [15], the types of knowledge with the most favourable cost/benefit ratio are quality constraints, attribute exclusion constraints, normality information, ordinality information and especially the derived attributes.

For example. the reason why the attribute exclusion constraint was included among the best rated attributes could be explained in the following way: it is relatively easy for an expert to mark some attribute as worthless for mining and excluding one attribute (particularly if it has many possible values) can dramatically reduce the number of combinations which need to verified or generated in subgroup or association rule mining algorithm.

As a conclusion, the findings of [15] when properly verified on the current domain and DM algorithm can help the analyst to decide which knowledge should be acquired weighting the

costs on one side and benefits on the other. For instance, if the data analyst has to decide, whether to acquire similarity or ordinality[24], the rule of thumb advises to acquire the latter. In any case, [28] presents experimental results that proper preprocessing can not be completed without the use of additional information on both website the content and structure of the website.

[24]Ordinality information may be for example used to exclude all combinations of values for a given attribute, where the values are not neighbours according to the ordinality information.

Chapter 3

Data Representation

The selection of the specific data preprocessing technique to be used is dependent on the requirements on the input data. Different data-mining methods require different data representations and work with different pieces of information. This chapter focuses on a specific issue which often arises in a WUM scenario – the need to reduce the dimensionality of input data. The goal of this chapter is to present the thesis author's contribution to dimensionality reduction in WUM applications, which is presented in the last two sections. The reason is that first a thorough definition of the problem and a review of the existing approaches need to be made.

Generally, there are two primary sources of information which need to be combined during the data preprocessing stage in order to construct the *visitor profile*. Visitor profile is the final representation, which is used as the input for datamining algorithms.

The two primary sources of information for visitor profile construction are the visitor model and the document model. The visitor model describes the clickstream of the visitor including the pages viewed, their order and the time spent on them. The document model describes the pages on the website. Its typical features are tokenized URLs, words on the page and their weights, the URIs reachable from the page, and mapping of the page to some taxonomy or ontology.

These features are sometimes referred to as *modalities*. Many modern approaches are based on building visitor profiles, albeit there is no widely accepted standard of what information should be contained in the visitor profile and how it should be represented. For example [25] uses vectors to represent modalities and a multi-modal vector to represent the entire model. In the following text, we will assume that input data for data mining algorithms are instances represented by vectors.

Even a shallow coverage of the preprocessing steps required by the WUM algorithms referenced in the survey [33] is beyond the scope of this chapter. Instead, an attempt is made to divide data mining algorithms into two classes when it comes to their requirements on input data – depending on whether the instances need to be described using a constant number of attributes or not. In the further text we will refer to the first class as *fixed-length input* and the second as *variable-length input* algorithms.

Of course, many other possible taxonomies of DM algorithms in relation to their requirements on input data can be invented. For instance some algorithms work only with nominal attributes while other work with numeric attributes. However, during the work on the empirical part of this thesis, it was found out that many algorithms used in WUM are *fixed-length*

input, while the nature of data is mostly variable-length. Efficiently coping with this discrepancy poses a significant challenge. Therefore, this chapter deals almost exclusively with this topic, which is also closely connected with the way how the visitor and document models are combined.

The remainder of this chapter is structured as follows. Section 3.1 introduces fixed-length input algorithms, Section 3.2 introduces variable-length input algorithms and Section 3.3 introduces the curse of dimensionality. A taxonomy of existing solutions for dimensionality reduction is given in 3.4, Section 3.5 details feature selection algorithms. In the following two sections, the thesis author extends his work on feature-extraction based approach for dimensionality reduction. In Section 3.6 several new WUM-specific attributes are proposed. In Section 3.7 the author suggests a new classifier-based approach for dimensionality reduction of web usage data.

3.1 Data Representation For Fixed-Length Input Algorithms

In mathematical terms, the fixed-length input class of algorithms require that the dimension of all the input vectors must be the same. The algorithms with this requirement include some of the most widely used ones including many association rule mining, clustering, neural networks and self organizing maps algorithms. This restriction is severe, because it is usually desirable that one vector typically represent one session on the web site.

Before the implications of this restriction are discussed, it is necessary to introduce some formalisms. The website w has c distinct pages $p_1, p_2, \ldots, p_c$. Random visitor session r on website w comprises of n_r pageviews, out of which m_r are distinct pages, $c >> n_r > m_r$. If the following equality holds

$$m_1 = m_2 = \ldots = m_n \text{ where } n \text{ is the total number of sessions in the data set} \tag{3.1}$$

it would be possible to express every session as vector a of length $d = m_n$. All the sessions could be represented as PageView matrix $PW_{n \times d}$

$$\begin{bmatrix} pw_{11} & pw_{12} & \ldots & pw_{1d} \\ pw_{21} & pw_{22} & \ldots & pw_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ pw_{n1} & pw_{n2} & \ldots & pw_{nd} \end{bmatrix} \tag{3.2}$$

The element $PW[x, y]$ denotes e.g. the URL of the resource viewed during the x-th pageview of session y.

The PW matrix is a sufficient input for fixed-length input algorithms. Unfortunately, it is highly unlikely that the equality 3.1 will hold for any website. Some visitors get on the website by error and view only one page before they leave the website, some explore almost the entire website before they leave it.

The basic approach used in WUM to deal with this problem is to remove the temporary information i.e. the order in which the pages appeared in the clickstream. Vector of dimension c then suffices to represent the pages viewed during a session. We call this vector *page subvector* of the visitor profile.

The elements of an individual page subvector $(w_1, w_2, \ldots, w_c)$ contain the weights of pages $p_1, p_2, \ldots, p_c$ as visited in the session. This representation leads to sparse vectors, because

ID	VID	tranID	TimeStamp	Entrance	AlpsTrip	LondonTrip	Search
1	1	sess_1	2007-JAN-01 9:00:00	30	0	0	0
2	2	sess_2	2007-JAN-01 15:05:01	0	30	0	0
3	1	sess_3	2007-FEB-01 15:10:38	0	0	30	0
4	2	sess_4	2007-FEB-01 23:12:00	15	0	30	45

Table 3.1: Vector representation of four sessions. The web site consists of four pages Entrance, AlpsTrip, LondonTrip Search. The weight in the session subvector is the time spent on page with 30 being the default value for last page in the session when the time on page is unknown. "Vis.ID" stands for VisitorID, "sess" for session and "tranID" for transaction ID.

ID	items	transactionID	visitorID
1	entrance	session_1	1
2	AlpsTrip	session_2	2
3	LondonTrip	session_3	1
4	entrance, search,LondonTrip	session_4	2

Table 3.2: Sparse representation of the columns 1, 7-10 of the binary matrix 3.3

almost all vector elements for a given session usually contain zeros as a result of $c >> n_r$ assumption made earlier.

It is convenient to omit the zero elements and use a more efficient *sparse representation*[1] of the vectors such as in [43]. If we are not interested in the weights, but only in the presence of a particular page in the session a *binary incidence matrix* as exemplified on Table 3.3 may be used to represent the sets of visited pages. The sparse representation of the columns 1, 7-10 of this binary incidence matrix is shown on Table 3.2.

Unfortunately, as apparent from the example, this representation can not be also directly used for fixed-length input class of algorithms. However, the sparse representation is due to its efficiency often used for data persistence and manipulation (for example in the Arules package [43]). The sparse representation needs to usually be "decompressed"' back to the very inefficient transaction representation before the data are sent to the datamining algorithm.

Some algorithms require that all attributes are binary. The approach to binarizing an attribute depends on the type of the attribute. The easiest situation is apparently with boolean attributes, because they require no additional preprocessing. If the attribute has too many possible values, which is usually the case with continuous numeric attributes (e.g. product price) and sometimes even with linear discrete attributes it is necessary to carry out discretization and represent the whole value range by several intervals. It is conceivable that some nominal attributes may also need merging of several attribute values into one value due to excessive number of categories. After all attributes are represented by a reasonably small number of values, it is possible to carry out the binarization.

Binarization is a process of transforming a nonbinary attribute into a set of binary attributes. The example result of binarized data is the binary incidence matrix on Table 3.3.

[1] In some sources (e.g. in [43] a distinction is made between a *sparse vector* and a *sparse representation* of a vector. A sparse vector has null values in most of its components. Sparse representation of a vector lists only the components of the vector, which do not contain nulls.

ID	JAN	FEB	Mor	After	Night	Entrance	AlpsTrip	LondonTrip	Search
1	1	0	1	0	0	1	0	0	0
2	1	0	0	1	0	0	1	0	0
3	0	1	0	1	0	0	0	1	0
4	0	1	0	0	1	1	0	1	1

Table 3.3: Transaction data from Table 3.1 represented as a binary incidence matrix. Transaction ID and visitor ID were omitted for space reasons.

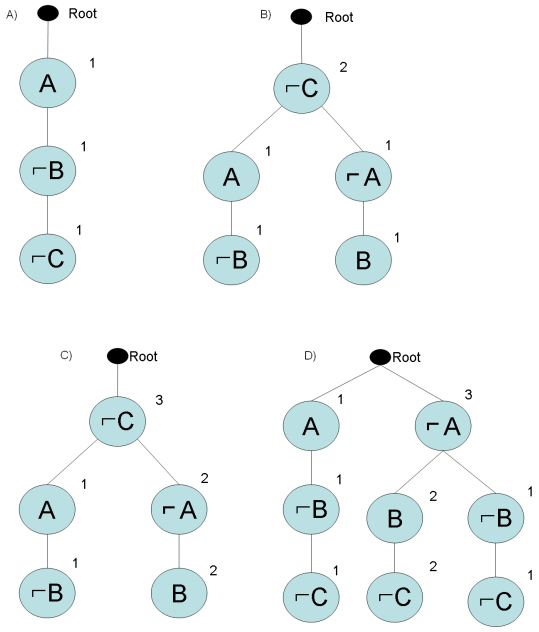


Figure 3.1: Step by step construction of a prefix tree based on Morning, Afternoon and Night columns of Table 3.3

Binarization as outlined above is commonly used, but the empirical results from the case study part of this thesis presented in Section 7.5 indicate that widely used data mining algorithms had very poor time performance on real world web usage data from a business website when preprocessed in the aforementioned way. The reason is so called "curse of dimensionality"', which is described later in this chapter in Section 3.3.

Internally, some data mining algorithms use a different representation for performance reasons. For example the latest versions (the functionality added in version 4.03 [19]) of the arule package uses a prefix tree to process transactions with the same prefix together [20].

An in-depth analysis in using a prefix tree in GUHA-style association rule mining algorithm is given in [84]. This idea was probably first introduced in a relatively young algorithm called *FP-growth*. A results of a comparative study [38] indicate that FP-growth is one of the fastest association rule mining algorithms. Figure 3.1 gives an example of a prefix tree construction.

visitorID	Visitor sequence
1	<(entrance)(LondonTrip)>
2	<(AlpsTrip)(entrance Search LondonTrip)>

Table 3.4: Session sequence version of the example data

3.2 Data representation for variable-length input algorithms

So far we have focused on the preprocessing for fixed-length input algorithms (such as association rules[2]). The variable-length input algorithms as represented by sequential pattern mining algorithms such as GSP or AprioriALL are more relaxed when it comes to the requirements on the representation of the input data.

The two important notions in mining for sequential patterns are an *itemset* and a *sequence*. An itemset is a non-empty set of items, the itemset i is denoted by $(i_1 i_2 \dots i_m)$ where i_j is an item. In WUM items are usually page views/user actions/events. A sequence is an ordered list of itemsets. A sequence s is denoted by $< s_1 s_2 \dots s_n >$, where s_j is an itemset.

Sequential algorithm can be useful when we want to analyze patterns with respect to the order of the visited pages. If we restrict the analysis to a session level, an itemset might contain a single item – the visited page, and a sequence might correspond to one session.

When analyzing the visitor behaviour across multiple sessions, the sequence comprises of itemsets containing all pages visited during each of the visitor's sessions. Note that in the example that we use throughout this chapter there are two visitors each with two sessions. Table 3.4 represents this information in the sequential patterns format as introduced by [9].

Some implementations of sequential algorithms impose similar restrictions on input data as association rule mining algorithms. For example, based on the available published sources, the approach [22], which uses sequential patterns to analyze manufacturing warranty data, is severely limited what concerns both itemsets and sequences:

The itemsets may contain maximum two items – two labour codes representing the types of repair done during one warranty claim. Sequences may contain maximum two itemsets - *condition* and *decision* itemset.

The condition itemset contains up to two labour codes c_1, c_2 concerning repairs pertaining to the first car failure, while the decision itemset contains again up to two labour codes d_1, d_2 pertaining to the second, consecutive car failure. The sequential pattern is represented as an IF-THEN rule, where the body of the rule is the condition and the head of the rule is the decision.

In addition, the authors work with repair cost and the time span between the first and consecutive car failure.

It is interesting that such a simplified approach may work in practice. The authors applied their algorithm on 680,000 records of warranty claims for a one car model with the total warranty costs over 80 million dollars. Each object was described by 88 attributes. There were more than 2,200 labor codes. On such a substantial amount of data of considerable dimension a very long running time of the algorithm could be expected. However, under

[2] Although association rules are usually applied on market basket data, where there are naturally a variable number of items in the basket, some AR implementations require the fixed-length representation at least internally. This is the case with Arules Apriori algorithm see Arules documentation[43] as well as with the 4ft-Miner's bitstring approach (see Section 4.3.1).

visitorID	S1 P1	S1 P2	S1 P3	S2 P1	S2 P2	S2 P3
1	entrance			LondonTrip		
2	AlpsTrip			entrance	Search	Londontrip

Table 3.5: Processed data for sequence analysis, S1 and S2 denotes first and consecutive session, P1,P2,P3 denotes first, second and third pageview during a session.

the least favourable setting the running time of their algorithm was below 400 seconds (no description of hardware used is given).

It is probably the simplification of the problem representation, which lead in their case to the substantial performance boost. It is a question what was the extent of the inevitable loss of information caused by this simplification. It could be only guessed that in most cases, the repair did not involve more than two labour codes and that a previous car failure(s) lead to another product fault(s) only once at max. In a different scenario, increasing the permissible length of condition or decision itemset is conceivable as well as prolonging the sequence by several other itemsets.

Due to the very brief nature of the paper [22], it is possible that there are some misinterpretations in the aforementioned text. However, this example was selected only to illustrate that an efficient yet imprecise representation may be worth a consideration.

In addition, the sequence data are here represented in a constant number of attributes. This data representation meets the stricter requirement of fixed-length input algorithms. The condition itemset represents the attributes permissible in the body of the association rule and the decision itemset the attributes permissible in its head.

In some situations it might be possible to use this approach to dimension reduction for clickstream representation. The simplifying assumptions here are that one visitor may have maximum a certain number of sessions and each session may comprise of certain maximum number of pageviews. These maximum numbers would probably need to be reasonable small, because the datamining algorithm may not perform well when most instances have a null value in a given attribute. Both parameters can be determined using descriptive statistics.

In our example the average number of sessions per visitor is 2 with a standard deviation 0. The average number of page views per session is 1.5 with the standard deviation of 0.87. Based on this data, it is reasonable to set the maximum session length to 3. The preprocessed input would then be represented as indicated on the Table 3.5.

There is only a small step from this representation to the matrix 3.2. There are, however, many apparent reasons which make this simple approach infeasible in practice in most cases. A more feasible alternative is e.g. presented in Section 3.6.

3.3 Curse Of Dimensionality

The success and the low time complexity of the analysis of manufacturing warranty data described in the previous text could be attributed to the fact that the authors could represent a highly dimensional space with 80 attributes by only four attributes. Such transformations are very useful when overcoming the *curse of dimensionality*, which is a well known and frequently run into problem in data mining and statistical engineering when dealing with high-dimensional data.

The curse of dimensionality essentially means that

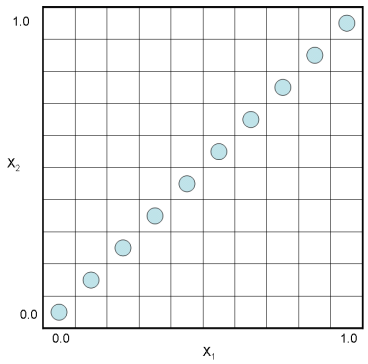


Figure 3.2: Ten points on 10x10 grid, the points are not drawn at random to illustrate their decreased coverage.

> the amount of data to sustain a given spatial density increases exponentially with the dimensionality of the input space, or alternatively, the sparsity increases exponentially given a constant amount of data, with points tending to become equidistant from one another [75].

The same source [75] notes that generally the curse of dimensionality will adversely impact any method based on spatial density, unless the data follows certain simple distributions.

A very fitting explanation of this phenomenon accompanied by graphical illustration on a multidimensional grid is given by Charles Annis [11], we will paraphrase it in the following paragraph.

Lets consider two independent variables both defined on $(0, 1)$ interval and both represented by ten samples along that interval. Each sample represents in average 10% of the probability space. Taking these two variables as coordinates of points in a two dimensional space, we get a new probability space of $10 * 10 = 100$ area units. As a consequence, each of the ten points now represents only 1% of the probability space as seen on Figure 3.2.

Curse of dimensionality is well known issue in Neural and Bayes networks. According to [11] it places a practical limit of about a half dozen dimensions on direct-sampling Monte Carlo. To quote the StatSoft Electronic Textbook [72]:

> In practice, the complexity of neural networks becomes unmanageable when the number of inputs into the neural network exceeds a few hundreds or even less, depending on the complexity of the respective neural network architecture.

Finally, the curse of dimensionality also applies to association rules [47]. This paper illustrates it on the typical application area for association rule – market basket analysis: if a retailer offers 10.000 different products, one arrives to the overwhelming $2^{1000} \approx 10^{3000}$ different market baskets. Limiting the number of possible items to 30 still results into a difficult problem with more than 10^{100} possibilities. Although the association rule mining algorithms are build to circumvent this so called combinatorial explosion, mining very long itemsets is not mostly feasible even when various optimizations are in place.

For example *apriori* [8], the most cited association rule algorithm, needs to check about one million subsets when generating itemsets of length 20. A more detail explanation of this problem is out of the scope of this chapter, but is available e.g. in [47]. The same source claims that

Any data mining algorithm will have to somehow deal with this curse.

There are several possible solutions to this problem. Many of them are based on reducing the *feature space* i.e. cutting down the number of attributes used in data mining.

3.4 Dimensionality Reduction

In the previous section it was shown that for high dimensional data the feature space needs to be reduced. There are two basic alternatives how to approach this task, both can be done manually by an expert or algorithmically:

- Feature selection: select only the subset of available attributes
- Feature extraction: merge several existing attributes into a new single one

3.4.1 Feature Extraction

In mathematical terms, the *feature extraction* problem can be stated as follows: find a lower dimensional representation $y = (y_1, y_2, \ldots, y_M)^T$ of the N-dimensional random variable $x = (x_1, x_2, \ldots, x_N)^T$, where $M \leq N$ that captures the content in the original data as closely as possible according to some criterion [35, 2]. The components of y are sometimes referred to as *hidden components*. Another terminology note pertains to the term variable – in different fields synonyms "attribute"' and "feature"' are used.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \text{ feature } \underset{\rightarrow}{\text{extraction}} \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ y_M \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}\right) \tag{3.3}$$

There are two major ways of dimension reduction: linear and non-linear. Linear methods obtain the individual components of the new variable by linear combination of the the original variables.

$$y_i = w_{i,1}x_1 + w_{i,2}x_2 \ldots + w_{i,N}x_N \text{ for } i = 1, \ldots, M \text{ or } y = Wx \tag{3.4}$$

This process can be also expressed using matrices.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \text{ linear feature } \underset{\rightarrow}{\text{extraction}} \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ y_M \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \ldots & w_{1N} \\ w_{21} & w_{22} & \ldots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \ldots & w_{MN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \tag{3.5}$$

The specific representation of function *f*, which basically outputs the linear transformation weight matrix $W_{M \times N}$ is a goal of numerous methods. Paper [35] lists following linear dimension reduction techniques:

- Principal component analysis
- Factor analysis

The two most frequently used methods listed below are based on second order statistics.

In the case of variables with zero mean, the covariance matrix contains all the information about the data. For datasets where variables have non-Gaussian probability density distribution [74], higher order methods for dimension reduction are more appropriate. These methods include:

- Projection pursuit
- Independent component analysis
- Non-linear principal component analysis

PCA analysis is probably the most commonly used technique of dimensionality reduction. However, it may not be suitable for usually bulky web usage data because of its relatively high computational complexity. An interesting alternative to consider is the *Random projections* technique. This method have emerged in the context of clustering text documents, where the original data of dimension around 6000 are scaled down to the order of 100. Using a vector of important text features is a common method in Web Usage Mining algorithms. It is utilized for example in the Lumberjack method described in Section 7.2 of the case study part of this thesis.

In addition, it has been empirically shown that the results obtained with the random projection method are comparable with the results obtained with PCA while taking a fraction of the time PCA requires [35]. Another important property of this algorithm is that it preserves the similarity of data vectors well even when the data is projected to a moderate number of dimensions [35, 18].

Lets leave the discourse on feature extraction with the concluding remarks of [18]:

> random projection is a good alternative to traditional, statistically optimal methods of dimensionality reduction that are computationally infeasible for high dimensional data. Random projection does not suffer from the curse of dimensionality, quite contrary to the traditional methods.

3.5 Feature Selection

Feature selection is a relatively simple yet powerful method of dimensionality reduction. This corresponds to the finding presented in the previous chapter – attribute exclusion constraints belong to the most valuable types of background knowledge. Schematically, feature selection can be depicted as follows

$$\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_N \end{bmatrix} \overset{feature selection}{\longrightarrow} \begin{bmatrix} x_{i1} \\ x_{i2} \\ . \\ x_{iM} \end{bmatrix} \tag{3.6}$$

Although having more features should – at least in theory – result in more discriminating power, empirical results have shown that for many data mining algorithms this is not the case. data mining textbook [86] cites an experiment with a state-of-the art decision tree

learner C4.5, in which adding a random (*irrelevant*) attribute deteriorated the classification performance by 5 – 10%.

The performance of Naive Bayes is not deteriorated by irrelevant attributes, but should there be a *redundant* attribute on the input, the operation of Naive Bayes can be damaged.In some cases, even including an additional *relevant* attribute can be harmful [86]. Another strong reason for carrying out attribute selection is that fewer attributes usually result in a more easily interpretable representation of the target concept.

There are two fundamentally different approaches to selecting the attribute subset:

- *Filter method* – make independent assessment based on general characteristics of the data
- *Wrapper Method* — evaluate the subset by the same machine learning algorithms that will be used for mining

The problem pertaining to the filter method, which removes the least relevant attributes before the actual mining, is that there is no widely accepted measure of attribute relevance.

One method is to find a minimum set of attributes which can be used to discriminate between all available training instances. For example, two attributes would be probably too little – there would be multiple instances with the same value of these two attributes. At the other extreme, if, for example, a unique identifier of each instance remained in the training dataset by negligence, only this attribute would be selected and the model would overfit. This method has a strong bias toward the consistency of the attribute set.

A possibility is also to use a suitable machine learning algorithm (a different one than in the data mining phase) to select the attributes. Decision trees are particularly suitable for this task. Only the attributes used for the tree construction are used in the mining algorithm. An alternative is that the user may choose from the decision tree only the attributes with the highest information gain.

Yet another way of selecting the attributes is building a linear model and ranking the attributes based on the size of the coefficients. The algorithm used might be for instance a linear support vector machine [86]. A variant of this approach called *recursive feature elimination* [86] can yield better results with some performance sacrifice.

The methods described above only *rank* attributes, they do not provide a clue on how many attributes and which ought to be used. There are several methods which also draw the dividing line marking an attribute as "take"' or "not-take"'. One possibility is to use an analogy to near-hits/near-misses attribute weight learning method used in instance based learning (see e.g. [86]). After sufficient number of iterations only attributes with positive weights are taken. Unfortunately, this method fails to eliminate redundant attributes.

Most methods for attribute selection rely on fast search through the attribute space for the most suitable attributes. There are several methods, how to perform this search. Some of the more sophisticated ones involve genetic algorithms. These are based on the idea of natural selection. They usually operate with a population (a set) of individuals (attribute subsets). The suitability of the individuals is considered with respect to a given fitness function, only the best individuals survive and can reproduce. In section 3.7.4, a method based on Genetic Programming (GP), which is a kin of genetic algorithms, is introduced to deal with the curse of dimensionality.

3.6 Feature Extraction In WUM Applications

The current trend is to enrich the clickstream data with semantic information to build *visitor profiles*. Although undoubtedly highly beneficial, the acquisition of semantics can be relatively costly. In the Section 2.2.3 of the previous chapter, the study [15] has shown that derived attributes can provide excellent cost/benefit ratio. Generally, derived attributes are computed from the already available attributes, so they do not directly increase the knowledge acquisition costs. This does not mean that the computation of derived attributes does not utilize semantic information. In some cases, semantic information is indeed required, but it is assumed that it has been already acquired for some other purpose.

In this section, no automated generally applicable means of feature extraction such as PCA are introduced. On the other hand, several WUM specific derived attributes are proposed by the author of this thesis. Some of these attributes (*LandingPageRelevancy*) have never been to the author's best knowledge used, some were already utilized in data-mining in the authors previous work (*Path summarization attributes*) and some are adaptation of work proposed in research papers (*User behaviour attributes*). Replacing original by derived attributes in the *visitor profile* has been found to perform well in the author's previous work [53, 51, 52, 54].

User Behaviour Attributes

Derived attributes are particularly useful for measuring the user behaviour. User behaviour on an aggregate level is described in [87] with the following four characteristics:

1. *Backtracking tendency:* likelihood of backtracking to visited pages
2. *Exploration tendency:* likelihood of exploring the whole website
3. *Errancy:* likelihood to going to a wrong page during browsing
4. *Time:* likelihood of spending a long time on a web page

No semantic information is generally needed to infer these four characteristics. The range of values of these characteristics for a given website is $low, average, high$ or $-1, 0, 1$ respectively. For web mining purposes we are not usually interested in aggregate *likelihoods* but rather in these characteristics on an individual level as proposed by the thesis author[3]:

1. *Backtracking:* extent in which the visitor backtracked
2. *Exploration:* proportion of viewed pages on the total number of site's distinct pages
3. *Error rate:* number of times the visitor clicked on a link on page A, transfered to page B and than during very short time interval t backtracked to page A.
4. *Time:* the proportion of pages on which the visitor stayed longer than $p\%$ (e.g. 75%) of other visitors who viewed the page.

If semantics is available, additional derived attributes may be defined.

[3] Other definitions are of course possible.

Table 3.6: Clickstream of one visitor described in two semantic dimensions. The values in the Score column were computed using the ad-hoc version of the formula

O	URL	Type of trip	Country	Time	Score
1	NorwayTrip.htm	Adventurous	Norway	60	60
2	AlpsTrip.htm	Leisure	Austria	120	203
3	SkyingAustria.htm	Adventurous	Austria	240	504

Path Summarization Attributes

Section 3.2 presented an approach to dimensionality reduction based on representing a session by a fixed number of pageviews. One of the disadvantages of this approach was that if a session had more than the fixed number of pageviews, the remaining ones were omitted, if it had less, inconvenient nulls appeared in the profile for the remaining attributes. The approach presented here is more sensitive. Two groups of attributes are used to represent the path: *Important points on the path* and *Attributes conceptualizing the path.*

Important points on the path contain the *Entry page*, *Exit page* and *Conversion page.* Each point can be characterized by a vector containing information on the page such as a relevant concepts from some ontology and an expression of visitor's interest in each of the semantic dimensions of the page e.g. by using Score as defined by equation 3.7. For simplicity sake, a common Score value may be used for all the dimensions.

Attributes conceptualizing the path are more holistic in an effort to capture the "essence"' of the session. The *Range of interest* attributes express how much was the visitor focused in each of the semantic dimensions on a given granularity level. Given the example clickstream depicted on Table 3.6, the *Range of Interest* of both dimensions is 2 – there are two different attribute values in the *Type of dimension* (*Adventurous* and *Leisure*) and two different attribute values in the *Country* dimension (*Norway*, *Austria*). Perhaps the most important ones are the *MainInterestInDimension* attributes, which are based on a classifier-backed attempt to determine the visitor's interest in each of the dimensions on a given granularity level. This notion is in detail explained in Section 3.7.

In addition, Score of top-level concepts in both service and content ontologies may be used as attributes. For instance, the top-level concepts in the Service ontology might be *Fulltext Search*, *Directory Search*, *Product Page*, *Supporting Information.* The values of these four attributes could be determined based on the sum of Score of those pages in the session which were mapped to the respective ontology concepts.

Landing Page Relevance To Search Engine Query

Another example attribute *LandingPageRelevancy* could be formed based on full texts of visited pages and the visitor queries. An increasing number of visitors use search engines to navigate through web. The queries submitted to the search engine are very information rich. In fact, these queries are often the most explicit expression of the visitor's interest. Therefore, incorporating visitor queries into WUM is a very useful and promising are of research, which has been left so far largely unexplored.

The Search Engine Result Pages (SERPs) for a given term t usually comprise of the most

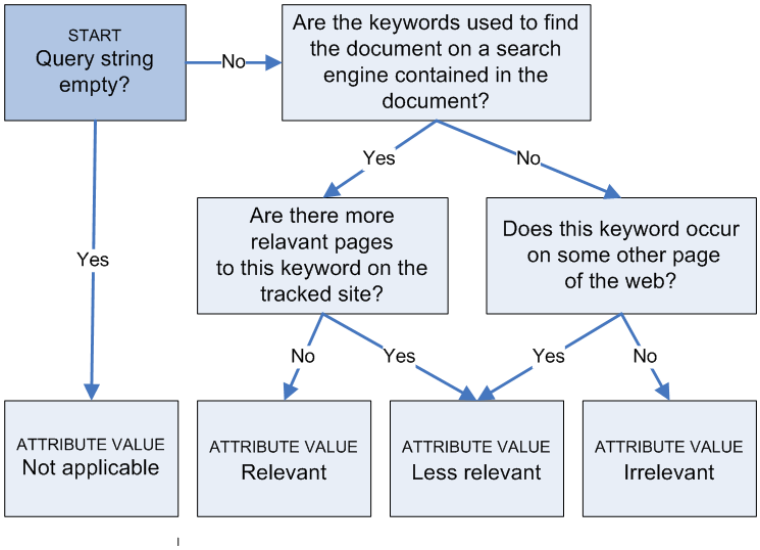


Figure 3.3: EntrancePageRelevancyToVisitorQuery attribute - Decision tree

relevant[4] pages to the query from the search engine's index. Search engines strive for diverse SERPs, so if one website includes multiple relevant pages for a given query term, only the most relevant page p_L is usually included in the SERPs. If the user clicks on the link to this page on her SERP, she enters the site from the page p_L or she *lands* on it hence *Landing page*.

The goal of the proposed *LandingPageRelevancy* attribute is to check, whether the search engine selected the most appropriate page from the tracked site for a given query term. This check is a non-trivial operation, which requires using information retrieval techniques to carry out the tasks depicted on Figure 3.3. This effort should be worthwhile, because it is a widely acknowledged fact that serving the right landing page can significantly influence the conversion rate.

The construction of the *LandingPageRelevancy* attribute is not inherently dependent on the availability of semantics, although some background knowledge might be beneficial. It relies on obtaining the actual content of the website. This is a relatively easy and routine task, which can be performed automatically with little costs.

A more challenging task is the preprocessing of the content and query terms; especially in the case of non-English websites. The most important step during the preprocessing is the removal of stop words and stemming or lemmatization. For many inflexional languages (e.g. Czech), there is nothing like the Porter Stemming Algorithm [65] for English. Well performing stemming algorithms for inflexional languages are usually so complicated that they need to be provided on a component basis by a third party[5].

Another difficult task is choosing the appropriate information retrieval algorithm. These might range from TF-IDF to more sophisticated techniques using different weights according

[4]Relevancy here is rather vague term as every search engine uses a very different and complicated algorithm for ranking. In addition, modern search engines often keep track of on which SERP links people click on and use this information to update page rankings. There is also the influence of other websites on SERPs through "votes"' computed by PageRank and similar algorithms. In addition, some search engines also put a pseudo random element into ranking as well.

[5]E.g. for Czech, there is at least one free program, which can be used for this purpose – the Free Morphology, and several commercial ones offered by for example by Lingea.

to the embedding HTML block, and even employ information contained in the the structure of the web by using link anchor text and or a HITS [50]/PageRank [62] type of algorithm. Generally, the more complicated the algorithm, the closer the ranking it produces should get to the algorithms used by search engines. Building such a complicated algorithm for constructing one, though very useful, attribute would be a waste of resources[6].

Considering the purpose of the *LandingPageRelevancy* attribute, a more reasonable approach would be to build a site specific algorithm based e.g. on TF-IDF which would use some background knowledge for assessing which page would be most suitable as a landing page for a given query term. If the principles of Search Engine Optimization (SEO) are taken into account during site design, it is easy to infer for which terms the page was optimized[7]. One of the ground stones of SEO is to put the key terms into the Meta description tag, page headings and URL.

The proposed attribute has four possible values with the following meaning:

Not applicable The visitor did not come from a full text search engine or the query string is empty.

Relevant The search engine returned the visitor probably the most relevant document from the tracked site matching the visitor's query.

Less relevant There are possibly more relevant pages to the query on the tracked site than the one returned by the search engine.[8]

Irrelevant There might be no pages relevant to the user query.[9]

3.7 WUM-Specific Classifier-Based Approach To Dimensionality Reduction

In the previous section, the advantages of using feature extraction and in particular Page Summarization Attributes were highlighted. In this section, a novel method of reducing the data dimensionality for WUM is proposed. This method was specifically designed as a preprocessing step for association rule mining algorithms described in Chapter 4 and demonstrated in Chapter 7. This section discusses how to compute Score – the weight metric used to construct many of the Page Summarization Attributes.

Score is a weight, but it can be also used to determine the prevalent value for an attribute. Although it may not appear at the first glance, Score can be used for classification as a *preprocessing* task. There is a vague analogy to using a decision tree to find the most relevant

[6]There are commercial tools available that could greatly facilitate this process. For example, Statistics TextMiner allows quick retrieval of documents from web as well as word ranking.

[7]In SEO, optimizing a page for a term means that the webmaster alters the page and links leading to it in an effort to persuade the search engine to determine that this page is highly relevant for a given keyword and place it on the first positions in the terms SERPs.

[8]This may happen because of a different algorithm the respective search engine uses to compute keyword-page relevancy or because of off-page factors (links from other pages) which may through increasing page importance outweigh deficiency in on-page factors.

[9]If a page is found on e.g. Google on a specific keyword, which is not contained in the page text, it is likely due to other sites/pages linking to this page with this keyword present in the link text. This phenomenon may indicate that the site is subject to Google Bomb, negative reputation or that there is an important relevant keyword omitted from the text.

attributes (see Section 3.5). Decision tree is there used to choose the most valuable attributes from the set of all attributes.

In contrast to decision trees, Score-based classification works "on instance"' basis. It is used to select the most appropriate attribute value for a given session[10] attribute. How is it possible that there are multiple attribute values to choose from for one session? Some of the attributes that describe a web page can be also used to describe a session. Session usually comprises of multiple pages, each can have a different value for a given attribute. Selecting one value out of the all the possibilities present for the attribute and session is a classification task.

In this section two ways to compute Score are presented. The first approach is a simple experimental heuristic formula, which was found to work well in one specific e-commerce data-mining scenario using association rule mining and clustering algorithms [53]. This formula is universal and no training occurs. The second approach is more exact and is based on training the classifier on a training set using genetic programming. It should be noted that Score is only a weight attached in the first approach to a pageview, in the second approach to attribute-pageview pair. The classification itself is done in the same simple way in both cases: weights are added up for individual attribute values, the value with the highest sum of weights is the result of the classification.

This rather general and theoretic description is illustrated on the following motivational example, which also contrasts Score-based classification to some other preprocessing techniques.

Consider an average e-commerce website with 1000 pages with available mapping of each of the pages to a service ontology and the effect granularity information may have on the preprocessing. For example, when we want to use association rules to investigate general usage patterns in relation to conversions, it may be enough to use the granularity information to "zoom out"' the data and use only the following attributes: the time spent on search related pages, the time spent on product pages, the time spent on general information pages. The head of the rule may be constituted only by conversion (a boolean attribute). The attributes in the condition need to be properly discretized e.g. into *none*, *low-to-medium* and *high* categories. In this way we would obtain a vector of dimension $3+1=4$. For comparison, representing the data on a finest granularity level for this site could easily lead to a dimension of 1000+, if the common approach to market basket representation illustrated on Figure 3.3 would be taken!

Unfortunately, many times it is not feasible to use low-granularity representation. One solution is to use high granularity representation, but only of selected aspects. For instance pages can be described on the semantic level with the "country"' and "type of trip"' dimensions, the former having 100 possible values and the latter 20. If the "type of trip"' is not necessary for the analysis, only the "country"' attribute can be used. This will result into 100 new attributes, each denoting the amount of time spent by the visitor on page tagged with a specific country, which is still rather many.

The backbone of the solution proposed in the following text is localizing the prevalent attribute value for the given attribute and visitor. For instance, the visitor's ideal holiday is of a leisure type and in France, and the fact that a trip to Sweden also appeared in the visitor's clickstream albeit with lower weight is discarded. Each dimension is represented only by the value with maximum sum of Score. It should be emphasized that in this preliminary

[10]It is possible also to modify it for working on a per-visit rather on per-session basis.

work there is one strong assumption – the visitor interest is focused only on one specific value of a given attribute. Other values even when present in the visitor clickstream are considered as navigational errors or substitutes.

3.7.1 Ad Hoc Version Of Score

The goal of the method described is to unveil the visitor preferences across each of the dimensions and use solely the value with the highest interest to represent the entire dimension. In our example depicted on Table 3.6 we have "country"' and "type of trip"' dimensions. In this approach, both dimensions can be represented on the finest granularity level with mere two attributes (as opposed to 120 in the usual approach).

There is this rationale behind the Score formula: the goal of the visitor is to find a product, which matches their preferences as much as possible. The visitor refines the search with each viewed page. Pages deeper in the visitors click stream can tell therefore more about the visitors target.

Score was designed to express the absolute weight of a particular page in the visitors path. Its formula takes into account time on page t ,a higher weight is given to pages with higher order number o. Following is an experimental ad hoc formula used to compute Score.

$$S = (\ln(o) + 1) * t \tag{3.7}$$

Score is computed for each pageview and is common to all dimensions of the page see the Score column of Table 3.6. The values with the highest sum of score in each of the dimensions are *Adventurous* with Score value $60 + 504 = 564$ in the *Type of trip* dimension and *Austria* with Score value $203 + 504 = 707$ in the *Country* dimension.

In addition to a considerable performance boost, there is a clear benefit during the interpretation phase – each visitor can be described in a crisp and short way, rather than in a fuzzy way by a long and mostly sparse vector. The apparent pitfall is the reliable determination of the visitor interest. There is no guarantee that Score formula 3.7 will give satisfactory results for every website and interest dimension.

3.7.2 Training The Score Formula – Motivation, Overview, Training Set

When a paper using the the ad-hoc version of Score was submitted to the ECML/PKDD-2006 Web Mining Workshop, one of the reviewers rightfully criticized the arbitrary character of the Score heuristic. The work introduced here is an attempt to overcome this objection. However, it is not a justification of the exact appearance of the Score computation formula 3.7. The newly proposed version even works with one more variable - the session order number. It can therefore, in principle, cope with multiple session scenarios.

The design of the theory behind the Score learning takes into account that the exact formula for Score is dependent on many hidden case specific variables such as site layout, site structure, the demographical characteristics of the site visitors etc. In addition, Score is no longer universal.

A dedicated Score formula can be trained for each of the dimensions – the "'average"' user interests also evolve across pageviews and sessions in each of the semantic dimensions differently. On a travel agency domain, consider a specific site where links on navigation pages contain the information about the type of the holiday described on the link's target page, but no information on countries visited is provided. Because the visitor knows the type

of holiday, but does not generally know the countries visited before she clicks on the link, the influence on time spent on page might be less indicative of the visitor's interest in the type of holiday than what concerns the countries she wants to spend the holiday in.

As a consequence, the proposed framework allows to determine a separate *interest function* for each of the dimension. These functions are not generic and not intended to work across multiple websites. On the other hand, this set of function should be determined for each website based on the training data.

The crucial problem with this approach is obtaining the training data. In contrast to the test data, the training data must contain explicit expression of the visitor's interest in each of the dimensions for which the *interest function* is to be determined. One solution, though not perfect, is to use the visits in which conversion occurs. For example, if the conversion means booking a trip, then the characteristics of the booked trip are the explicit expression of the visitors interest. It is important that the product description be consistent with page descriptions. In other words, interest function can be computed only for the semantic attributes/dimensions, which are used to describe both pages[11] on the website and the conversion (here the booked trip).

It is proposed to learn the interest function with *symbolic regression*, which is one of the major types of genetic programming applications.

3.7.3 Symbolic Regression

Symbolic regression tries to find a function f represented by a syntactic tree with predefined operations which best matches data from the training set [56]. Lets first contrast standard regression analysis to symbolic regression. Both methods operate on a training set in the form of a regression table $A_{train} = x_i/y_i; i = 1, 2, \ldots, n$ containing n points. The goal of standard regression analysis is to find optimal parameters w of the function $G(x; w)$ in order to minimize the following objective function:

$$E(w) = \sum_{i=1}^{n} |G(x_i; w) - y_i| \tag{3.8}$$

The adapted function $G(x, w_{opt})$ models the training set A_{train} .

$$w_{opt} = \underset{w}{argmin}\, E(w) \tag{3.9}$$

Unfortunately, standard regression is not useful in our case, because the function G is unknown. Instead, symbolic regression can be used to find the unknown function t. It is similar to standard regression in that it also minimizes the objective function (here absolute error)

$$E(t) = \sum_{i=1}^{n} |t(x_i) - y_i| \tag{3.10}$$

The minimum of this function is

$$t_{opt} = \underset{t \in \tau}{argmin}\, E(t) \tag{3.11}$$

[11] But not necessarily all pages.

The function t is sought using genetic programming. At first, a set of permissible operations (such as division, multiplication, sign, cosine, negation) has to be defined as well as maximum and minimum tree length. Using these operations random (but syntactically correct) syntactic trees are generated. The τ set used in the formula 3.11 is a set of all non-equivalent syntactic trees, which can be generated. In other words, τ is a set of all permissible functions.

A tree represents a candidate function. The value of the objective function 3.10 is computed for all members of the A_{train} set. Then the "natural selection" operator is applied. There are several possible implementations of this operator.

In general, only the best individuals with the lowest values of the objective function survive and can reproduce. The reproduction process composes of two important operators which make the genetic programming generally much faster than blind search through the space of all possible syntactic trees. These "miraculous" operators are the mutation and crossover. The mutation operator randomly replaces a subtree of the syntactic tree by a random subtree. Both subtrees have to be syntactic correct and must meet certain length criteria. The crossover operator mimics exchange of genetic information by swapping two randomly chosen subtrees between two syntactic trees.

3.7.4 The Problem Formalization For GP Solver

Now, it can be proceeded to the formalization of the problem in a mathematical notation. First, it is needed to slightly redefine the training set. The training set A_{train} consists of N_{tr} instances, $N_{tr} \leq C$, where C is the total number of clickstreams of converted visitors available.

Next, each instance $a \in A_{train}$ represents *one visitor*. Training instance a is a multimodal vector. It is composed by a one-dimensional vector containing the ID of the instance and vectors t, n, r, O. the elements of A_{train} are identified by the ID of the training instance $v = 1, 2, \ldots, N_{tr}$. In the following text we will refer to the subvector x of the instance with ID v as x_v. In this way, for a training instance with ID v we get vectors t_v, n_v, r_v, O_v. A notation $x_v(i)$ is used to refer to the i-th component of the subvector x_v.

The vectors p_v, t_v, n_v, r_v represent the visitor's clickstream. The elements of these vectors correspond to individual pageviews $x_v = (x_1, x_2, \ldots, x_i, \ldots, x_c)$. Index i of the elements in these vectors denotes the order in which the individual pageviews occurred in time in ascending order. The number of elements c corresponds to the total number of pageviews in the visitor's clickstream[12].

The vector p_v contains unique identifiers (e.g. URLs, IDs) of pages, the vector t_v denotes the time length of pageview (e.g. in seconds), the vector n_v denotes the ordinal number of the session into which the pageview belongs, the vector r_v denotes the order of the pageview in the session n_v.

Class labels are contained in the vector $O_v = (l_1, l_2, \ldots, l_n)$, where class label l_i pertains to the i-the dimension and n is the number of dimensions for which the interest function should be computed. *There is an assumption that the dimensions are independent.*

For example, take a visitor no. 567 who had two sessions, during the first one there were three pageviews in the following order: 50 seconds on page 510, 15 seconds on page 120 and 30 seconds on page 1, while the second session comprised only of one pageview: 30

[12] If the visitor had more than one session, than this is the number of pageviews across all the sessions.

seconds on page 1. The vectors p_v, t_v, n_v, r_v will look like follows: $p_{567} = (510, 120, 1, 1)$, $t_{567} = (50, 15, 30, 30)$, $n_{567} = (1, 1, 1, 2)$, $r_{567} = (1, 2, 3, 1)$. On our model web site we are interested in the *country* and *type of holiday* dimension hence $n = 2$. The element l_1 pertains to *country* and l_2 to *type of holiday* dimension. Consider further that the possible values of *country* dimension are *England, Ireland, Wales* and possible values for *type of holiday* are *Leisure, Adventure*. If our example visitor from the previous paragraph converted and bought a leisure trip to England, we get $O_{567} = (England, leisure)$.

Let set S_i contain all possible semantic descriptors in dimension[13] i. Also we define a function

$$s(p, d, q) = \begin{cases} 1 \text{ if the value of page } p \text{ in dimension } d \text{ is } q \\ 0 \text{ otherwise} \end{cases} \tag{3.12}$$

For example, let the page 120 be described with attribute values *Wales* and *Adventure*, then $s(120, 1, Ireland) = 0$ and $s(120, 2, Adventure) = 1$.

With the help of function s and set S_i, it is possible to proceed to the definition of the estimate of the class vector $\widehat{O}_v$ based on the available clickstream data described by vectors p_v, t_v, n_v, r_v.

$$\widehat{O}_v(i) = \underset{y, y \in S_i}{argmax} \sum_{x=1}^{c} f_i(t_v(x), n_v(x), r_v(x)) * s(p_v(x), i, y) \tag{3.13}$$

In order to evaluate the suitability of f_i the objective function needs to be defined. This requires measuring the distance between the class label from the training set $O_v(i)$ and its estimate $\widehat{O}_v(i)$. The baseline approach would be to return

- 1 if $O_v(i) \neq \widehat{O}_v(i)$
- 0 if $O_v(i) = \widehat{O}_v(i)$

However, a finer distance metrics might be useful when we consider the semantic nature of the attributes for which the interest function is learned. For instance, this metrics may differentiate between the class label of the country dimension "Norway" and the estimate being "Finland" or "USA", assigning lower value to "Finland" and higher to "USA" respectively. Therefore, the following generic dissimilarity function is introduced

$$DSim(c_1, c_2) \tag{3.14}$$

The specific implementation of this function is beyond the scope of this paper[14].

With the use of function $DSim$ and the estimate estimate $\widehat{O}_v$ we can express the objective function[15]

$$E(f_i) = \sum_{v=1}^{N_{tr}} DSim(O_v(i), \widehat{O}_v(i)) \tag{3.15}$$

[13]Here, we use terminology used in the Semantic Web. Lets remind that semantic descriptor is a synonym for attribute value and dimension is (roughly) a synonym for attribute.

[14]If the concepts are mapped to some ontology, semantic web related papers offer a variety of methods for measuring the distance between concepts e.g. [10].

[15]For simplicity sake, any penalization for excessive length of the function f_i was not incorporated into the formula. It is however common to penalize longer functions. By longer functions we mean here functions that are permissible but whose syntactic trees comprise more than a certain number of nodes. The penalization might be also progressive.

This objective function has a minimum

$$t_{opt} = \underset{f_i \in \Phi}{argmin}\, E(f_i) \tag{3.16}$$

By now, almost complete process of finding the function f_i has been explained. It remains to set the parameters of the genetic programming approach used. This task among other things determines the set of all permissible functions Φ. It should be noted that the objective function $E(f_i)$ does not need to be evaluated for all elements $f_i \in \Phi$. Genetic programming procedures are usually designed in a way that they stop after whatever of the following occurs first:

1. The value of the objective function drops below a certain threshold
2. Maximum number of iterations of the algorithm is reached

The solution yielded by a GP procedure is the function[16] f_i with the lowest $E(f_i)$. After all functions $f_i, i = 1, 2, \ldots, n$ are found, they can be used to replace the arbitrary Score formula 3.7.

[16] In some situations, there may be multiple functions with the same (lowest) value of $E(f_i)$.

Chapter 4

Data Mining Algorithms – Implementations

Web Usage Mining (WUM) is a quickly developing field of web mining. The basic method used by WUM is analysis of web logs, which are files containing the streams of clicks (i.e. visited pages) of all the visitors of a website in a given time period. The analysis is usually carried out by statistical techniques, widely available commercial and open source tools offer descriptive statistics, more sophisticated tools use OLAP and in scientific papers one can encounter a full range of advanced data mining techniques, the most frequent of which is generation of association rules [33] and clustering.

This chapter first introduces two distinct association rule mining algorithms GUHA procedure Assoc and Apriori with their respective implementations 4ft-procedure of the *LISp-Miner* system (GUHA) and Arules package of the R Environment (Apriori). Next, three variants of the popular K-means clustering algorithms are introduced.

This choice was not random, the same algorithms were used in the case study part of this thesis. This chapter together with Chapter 7, where these algorithms are demonstrated on real web usage data, should help the analyst to decide, which algorithms with which settings are suitable for a specific Web Usage Mining task.

Interest measures are the key setting for association rule mining algorithms. If applied during mining, they restrict the number of rules generated. This is typical for GUHA. However, the Apriori algorithm does not generally allow to use additional interest measures during mining, so they are more commonly utilized after mining to select the most interesting rules. This chapter also covers the topic of selecting the appropriate interest measure.

It should be noted that the association rule mining part of this chapter 4.1 – 4.4 focuses more on the systems implementing the algorithms, while the clustering part puts the emphasis on the algorithms. The reason is that the *4ft-procedure* and the Arules system are heavily optimized existing solutions which are built on very different algorithms. In addition, *LISp-Miner* does not have open source code. In contrast, the clustering part describes one-purpose implementations of three variants of the K-means algorithm as programmed by the author of this thesis.

It is acknowledged that this dual approach causes an inconsistency on one hand, on the other hand it hopefully allows for a greater variety. On a relatively short space, it is possible to describe and contrast two long developed data-mining systems from the user's view, as well as show the programmer's perspective on the implementation of an existing data-mining

algorithm and designing its enhancements.

This chapter is structured as follows. Section 4.1 contains an overview of the two association rule mining algorithms (GUHA and Apriori) and their specific *4ft-Miner* and *Arules* implementations. Sections 4.2 and 4.3 highlight interesting aspects of the individual systems: particularly the configuration options of *Arules* and the data mining algorithm of *4ft-Miner*. The part of this chapter on association rules is concluded with Section4.4 on the selection of the suitable interest measure. The clustering part is opened with the description of multimodal clustering and its distance metric in Section 4.5. Next, the following algorithms are discussed: Spherical K-means in Section 4.6, its hierarchical modification Repeated Bisection Spherical K-means in Section 4.7 and a variant of the latter with initialization with a genetic algorithm in Section 4.8. Section 4.9 gives some remarks on the selection of a data mining algorithm.

4.1 Association Rule Mining Algorithms

The Apriori algorithm was introduced by Rakesh Agrawal in 1993 and has been since then implemented in many commercial and academic systems. We chose the *Arules* implementation, which is a part of the popular free R data mining environment. Because the Apriori algorithm is notoriously known in the data mining community, it is not explained in great detail, the focus is rather on the features and extensions of its state-of-the-art implementation in the *Arules* package[1].

GUHA is an original Czech data-mining method, which has its roots in the 1960'. Out of its several implementations (*GUHA+-*[2], *LISp-Miner*[3] and *Ferda*[4]), the *4ft-procedure* of the *LISp-Miner* implementation is described here. LISP-Miner comprised of six GUHA procedures [66] which mine for variety of patterns. The 4ft-procedure implements GUHA Assoc procedure, which mines for generalized association rules.

Many other association rule mining algorithms are only slight modifications of the Apriori algorithm, but GUHA does not belong among them. The section devoted to *4ft-Miner* is mainly focused on its less known algorithm. It is absolutely beyond the scope of this chapter to describe its rich functionality, however, some remarks are made here as well as in Chapter 7. For more comprehensive information please refer to *LISp-Miner* documentation [3].

Because the terminology used in GUHA is very different from what is commonly used in the data-mining community, we present here a short glossary. Some abbreviations used throughout the text are introduced there as well.

4.1.1 GUHA glossary

The GUHA method has been developed largely independently on the research in association rules carried out by Agrawal and his successors, the terminology used in *LISp-Miner* (GUHA implementation) and *Arules* (an implementation of the Agrawal's Apriori algorithm) is quite different.

The names for individual statistical measures of interestingness (called quantifiers in *LISp-Miner*) are in *LISp-Miner* often derived from the naming conventions introduced in the GUHA

[1] `http://cran.r-project.org/src/contrib/Descriptions/arules.html`
[2] `http://www.cs.cas.cz/ics/pc/win/guha/`
[3] lispminer.vse.cz
[4] `ferda.sourceforge.net`

Apriori	LISp-Miner
Rule body (LHS)	Antecedent
Rule head (Consequent,RHS)	Succedent
Confidence	Founded implication (FUI)
Support	Relative Base
Item (Attribute-value pair)	Literal (approximately)
Statistical measure	Quantifier
Classical association rule	Relative Base + FUI
Jaccard coefficient	Double Founded Implication
2x2 contingency table	Four-field (four fold) table

Table 4.1: Apriori vs GUHA terminology

book [44].

Throughout the following text, we will sometimes use the term *item* when talking about *LISp-Miner*. This is not precise, the right notion is *literal*. This difference is not only in the name; *LISp-Miner*'s *literal* e.g. $A(a_1, a_2, a_3)$ is more general than *item* $B(a_1)$ within the classical association rule theory. As it can be seen from the example, literal denotes an attribute and its multiple possible values, while an item denotes always an attribute and one of its possible values.

4.1.2 Task Settings

As the design of a system determines its possible use, we will here briefly describe the properties of *LISp-Miner* and *Arules*, which the most marked influence mining results and performance. The parameters explained here are then demonstrated by an experiment in Section 7.3.

The settings of the mining determine the running time, the number of generated rules, their appearance, and fundamentally their quality. Although both the user interface and the underlying algorithms of *Arules* and *LISp-Miner* differ significantly, we found that they have a common "core" subset of parameters.

Rule Appearance

The rule appearance determines, which items can appear in the body, and which parameters can appear in the head of the rule. This is possible to set in both systems, however *LISp-Miner* offers a sophisticated system of grouping items into "partial cedents" and imposing constraints not only on LHS/RHS as a whole, but also on the individual groups. The possibility to specify LHS and RHS is in most AR systems, *LISp-Miner* offers the following extra features

- **Condition group of items** Condition acts like a filter – only rows satisfying the condition are left in the data matrix for association rule generation.
- **Mandatory items** Such an item must appear in the rule.
- **Gace type** specifies, whether the item can appear in its positive form (the only option in *Arules*) or in its negation.

- **Coefficient** is defined for each item. Its default option "subset 1-1" is equivalent to the item within the classical association rules concept. Other options allow to generate items (literals in LM) with more than value.
- **Literal connection options** LM allows to specify, whether the literals are joined with conjunctions (the only option in *Arules*), or disjunctions.

Length Constraints

The maximum length of the generated association rule is an important property, which has a strong influence on the running time of the algorithm. We observed that with increasing rule length the running time grows exponentially. Arules allows only to set the maximum length of the rule. In *LISp-Miner*, constraints for LHS and RHS of the rule are defined separately. For each partial cedent, a minimum and maximum length constraint can be specified.

Quantifier

In *LISp-Miner*'s terminology, quantifier is a condition imposed upon the relation between LHS and RHS of the rule. The only "quantifier" within the traditional association rules concept is a relation described by confidence and support.

Soon after Agrawal introduced his conception of association rules in 1993, it became clear that confidence and support are not strong enough to distinguish good (interesting) rules from bad ones. Many *interest measures* were introduced to solve this problem. These measures are however applied "ex post". At first, many rules are discovered using weak confidence and support thresholds and then they are sorted according to a selected interest measure(s).

Arules go mainly along the classical association rules concept, although they offer also a limited possibility to use one of the several interest measures in addition to confidence and support during mining.

Perhaps the biggest advantage of the bitstring approach implemented in *4ft-Miner* is that it enables mining not only for association rules based on confidence and support but also for rules corresponding to various additional relations of Boolean attributes including relations described by statistical hypotheses tests. Any combination of the interest measures offered by *LISp-Miner* can be used as a quantifier.

Rule Support

The notion of rule support deserves attention, as its (default) implementation in *Arules* differs from original Agrawal's approach (and *LISp-Miner*'s).

In Apriori as described by Agrawal [8], support is defined as the relative number of cases matching both the body and the head of the rule, which corresponds to the a frequency from the four-field table depicted on Table 4.2. Four-field table is sometimes referred to as Four-fold table or 4ft-table. In *LISp-Miner*, one can opt to use relative frequency or absolute frequency. However, support in *Arules* is defined as the frequency of the antecedent of the rule, which corresponds to the $a + b$ frequency from the 4ft table.

Borgelt, the author of the *Arules* AR algorithm, decided for this measure [20], because he believed that it is more compatible with the additional measures of interest, some of which are introduced later. He claims that this interpretation of support tells more about the statistical basis of the rule. Arules gives the option to use the original definition of support as well.

Table 4.2: Four field table

	Suc	$\neg Suc$	$\sum$
Ant	a	b	r
$\neg Ant$	c	d	s
$\sum$	l	l	n

LISp-Miner does not allow to use the $a + b$ frequency as a support threshold.

4.2 Arules

The *Arules* package interface a state-of-the-art association rule mining algorithm written by Christian Borgelt (University of Magdeburg), which is also used in the Clementine package available from SPSS.

4.2.1 Mining Algorithm

The Apriori algorithm [8, 86] is in the core of the Borgelt's implementation. The classical Apriori algorithm has two stages. In the first stage, all one-item sets with minimum coverage (support) are generated, using this intermediate result two-item sets are generated etc. In the second stage, rules are generated from these item sets. Only rules satisfying the minimum accuracy (confidence) are generated. Borgelt's implementation in addition checks if constraints raised by additional measures if defined are met.

4.2.2 Configuring Arules

The *Arules* package exposes Apriori method, which can be handed over three parameter objects: *ASparameter*, *APappearance*, *APcontrol*. The Apriori method can be used to mine not only for rules, but also for frequent itemsets and hyperedgesets[5]. Hereafter, we will focus only on parameters relevant to rule mining.

ASparameter defines support, confidence, `minlen` (minimal number of items per set), `maxlen`, `target` (type of association mined), `smax` (maximal support), `arem` (use of additional rule evaluation measure), `minval` (minimal value for measure from arem), `orginalsupport` (use LHS and RHS support instead of LHS support).

The possibility to define additional rule evaluation measure makes the *Arules* environment special compared to many other implementations of Apriori. The author of the mining algorithm in Arules gave the following rational for including this measure

> Potentially interesting rules differ significantly in their confidence from the confidence of rules with the same consequent, but a simpler antecedent. [19]

[5]association hyperedges are sets of items (i.e. not rules) that are strongly predictive w.r.t. each other. Sourcce `http://www.borgelt.net//doc/apriori/apriori.html#hyperedges`

The additional rule evaluation measure arem can have the following values:

- **Absolute confidence difference to prior** – confidence of the rule minus the prior confidence of the RHS[6]: $\frac{a}{a+b} - \frac{a+c}{n}$
- **Difference of Confidence Quotient to 1** – confidence of the rule divided by the prior confidence of the RHS: $\frac{a}{a+b} / \frac{a+c}{n}$. The result (or its reciprocal value whichever is smaller) is then compared with 1. The idea behind this measure is that an increment of the confidence from 30% to 40% is more important than an increment from 60% to 70%, since the relative change is greater.
- **Absolute Difference of Improvement Value to 1** – a symmetric variant of the preceding measure, defined as $\frac{a}{a+b} / \frac{a+c}{n}$. *Absolute Difference of Improvement Value to 1* (AIMP) corresponds to *4ft-miner's* Above Average Implication (AAI) Quantifier, but its value is for a given contingency table is slightly different: $AAI = AIMP - 1$.
- **Information Difference to Prior** – it is an analogy of the information gain criterion see e.g. [86].
- **Normalized χ^2 Measure** – the χ^2 measure is divided by the number of item sets, which results into possible values of this measure between 0 (no dependence) and 1 (very strong dependence).

The additional rule evaluation measures are always used in conjunction with support and confidence (the rule must comply with both of them). [19]

APappearance This parameter denotes which items can appear where in the rule. The possible settings are: Left-Hand Side of the rule only, RHS only, both LHS and RHS, none.

AScontrol These groups of parameters contain settings for sorting rules with respect to their frequency, filtering and various performance tunning settings.

4.2.3 Postprocessing rules

As explained earlier, a typical run of *Arules* procedure yields a large number of rules (easily more than a thousand). Arules therefore offers several possibilities to cope with this flood of rules – one can either cluster the rules or sort them according to one of the available interest measures.

When we experimented with clustering of the associations generated in the case study presented in Chapter 7, we ended up with unreadable results, so we decided not to develop the notion of clustering associations further.

The interest measures available in Arules *interestmeasure* procedure are *chiSquare*, *cosine*, *conviction*, *gini*, *hyperLift*, *hyperConfidence*, *improvement*, *leverage*, *phi*, *oddsRatio*.

[6] Prior confidence is as far as we understand it the percentage of the rows satisfying RHS in the data matrix, which should be equal to the support.

4.3 LISp-Miner

LISp-Miner is an academical mining software available free-of-charge from `lispminer.vse.cz`. Its *4ft-Miner* procedure mines for generalized association rules, a special kind of which are association rules as defined by [8]. In this section, we describe the *LISp-Miner*'s 4ft-Miner procedure and we will mention the *SD-4ft* procedure as well, because it is used in the *LISp-Miner* Experiment no.3 in Section 7.3.6. More details on SD4ft-Miner can be found e.g. in [66]and in the LISp-Miner documentation.

4ft-Miner mines for rules based on statistical hypothesis test including conditional association rules. SD4ft-Miner mines for *SD4ft-patterns* – couples of sets that differ significantly in respect to some association rule. *SD4Ft-pattern* basically means that the subsets given by Boolean attributes α, β differ in what concerns the relation of Boolean attributes φ, υ when condition γ is satisfied.

LISp-Miner has, unlike Arules, a relatively easy-to-follow GUI, which guides the user through preprocessing, mining and result visualization.

4.3.1 Mining Algorithm

The basic task of association rule mining algorithm is to find association rules, which satisfy the requirements imposed on them by minimum support and minimum confidence parameters. This requirement (and the entire notion of association rule) was introduced by Agrawal et al [8] and his now classical association rule mining algorithm *Apriori*.

However, Agrawal in a sense rediscovered the founded implication quantifier, one of the range of quantifiers introduced within the GUHA framework in the 1960'. The work of Agrawal soon became one of the most quoted papers in the whole field of database research, with closely 1000 citation according to Citeseer.

The LISp-Miner implementation of the GUHA Assoc procedure is stronger what concerns the possibilities to formulate the data mining task. Its first feature was already mentioned in Section 4.1.1. If we consider an association rule $A(\alpha) \wedge B(\beta) \rightarrow C(\gamma)$, then within GUHA α can not only be a single value as in the original `Agrawal`'s approach, but also a subset of possible values of attribute A.

The algorithm used is based on a "bit string approach", which is sometimes referenced as *granular computing* [66]. The basic principle is to represent each attribute by cards of its categories. If the attribute A has six categories, then it is represented in each row by a string of six bits. Out of the six bits, only the bit corresponding to the actual value of the attribute A in the row R is set to 1.

The cards of Boolean attributes α and β are used to compute frequencies for 2×2 contingency tables. The four frequencies - a, b, c, d of the contingency table are computed using bit-wise operations on the cards of bits denoted as C. For example, the a frequency, which can be also represented as $C(\alpha \wedge \beta)$ is computed in two steps:

1. Bitwise conjunction $C(\alpha)\dot{\wedge}C(\beta)$ is applied
2. The a frequency is computed by a bit string function $Count(\epsilon)$, which returns the number of values "1" in the bitstring ϵ

In the LISP-Miner implementation of GUHA, the bitwise operations $\dot{\wedge}, \dot{\vee}, \dot{\neg}$ are carried out by very fast computer instructions. Moreover, the following optimization is utilized:

If the LHS of the rule comprises of several literals $C(\alpha_1)$,$C(\alpha_2)$, ...,$C(\alpha_i)$ than i cards of conjunctions $\alpha_1 \dot{\wedge} \ldots \alpha_i$ have to be computed. The computation of the n_{th} card of bits ($n > 1$) is boosted by using the result of previous $(n-1)^{th}$ computation.

The algorithm is approximately linearly dependent on the number of rows of the analyzed data matrix.

4.4 Selecting The Right Interest Measure

Selecting the right measure of interestingness is a difficult task, given the number of measures available and the fact that they can result in substantially different evaluations of interestingness of 4ft-tables. This has serious implications. In *GUHA*, generated rules must meet minimum values of chosen interest measures. In *Apriori*, all rules satisfying support and confidence are generated. Usually many rules are generated, interest measures are then used to sort the rules, only the rules on the top of the list are usually evaluated by an expert.

According to a research paper "Selecting the Right Interestingness Measure for Association Patterns" by Tan et al [79] not all measures are equally good at capturing dependencies between variables.

In the experiment presented in [79], 10 measures were used to sort 10 example 2×2 contingency tables. Surprisingly, some measures gave quite opposite results e.g. one example table was ranked 1^{st} by one measure and 10^{th} by another measure.

The authors of the paper list several key properties one should examine in order to select the right measure for a given application domain. These properties together with a list of measures are depicted on Figure 4.1 as adapted[7] from [79].

Properties P1, P2 and P3 as defined by [64] are widely recognized [79] and should all be satisfied by a good measure M.

- **P1:** $M = 0$ if LHS and RHS are statistically independent
- **P2:** M monotonically increases with $P(LHS \wedge RHS)$ when $P(LHS)$ and $P(RHS)$ remain the same.
- **P3:** M monotonically decreases with $P(LHS)$ (or $P(RHS)$) when the rest of the parameters ($P(LHS \wedge RHS)$ and $P(RHS)$ or $P(LHS)$) remain unchanged.

None of the measures satisfying P1-P3 listed in the paper [79] is directly implemented in *LISp-Miner* as far as we were able to match the measures from *LISp-Miner* to those from the study. However, we inspected the mathematical properties of the Above Average Implication (AAI) Quantifier and came to the conclusion that it satisfies properties P1, P2 and P3. Construction of formal proof remains for further work.

This observation is confirmed by empirical results from the previous work [51], where it was concluded that the subjectively evaluated quality of rules obtained with AAI and support as a quantifier was significantly better than with conventional confidence and support. The AAI is also available in Arules under the name *Absolute difference of improvement to 1* (AIMP).

[7] For space reasons, Klosgen's measure was excluded. In the original work, the $P(A)$ denoted the support of the antecedent of the rule (LHS) $\frac{a+b}{n}$, $P(B)$ denoted the prior confidence $\frac{a+c}{n}$ and $P(A \wedge B)$ denoted the posteriori confidence $\frac{a}{a+b}$. For better consistence with previous text, we use the following symbols P(LHS), P(RHS), $P(LHS \wedge RHS)$

Symbol	Measure	Range	P1	P2	P3	O1	O2	O3	O3'	O4
ϕ	ϕ-coefficient	$-1\cdots0\cdots1$	Yes	Yes	Yes	Yes	No	Yes	Yes	No
λ	Goodman-Kruskal's	$0\cdots1$	Yes	No	No	Yes	No	No*	Yes	No
α	odds ratio	$0\cdots1\cdots\infty$	Yes*	Yes	Yes	Yes	Yes	Yes*	Yes	No
Q	Yule's Q	$-1\cdots0\cdots1$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Y	Yule's Y	$-1\cdots0\cdots1$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
κ	Cohen's	$-1\cdots0\cdots1$	Yes	Yes	Yes	Yes	No	No	Yes	No
M	Mutual Information	$0\cdots1$	Yes	Yes	Yes	No**	No	No*	Yes	No
J	J-Measure	$0\cdots1$	Yes	No	No	No**	No	No	No	No
G	Gini index	$0\cdots1$	Yes	No	No	No**	No	No*	Yes	No
s	Support	$0\cdots1$	No	Yes	No	Yes	No	No	No	No
c	Confidence	$0\cdots1$	No	Yes	No	No**	No	No	No	Yes
L	Laplace	$0\cdots1$	No	Yes	No	No**	No	No	No	No
V	Conviction	$0.5\cdots1\cdots\infty$	No	Yes	No	No**	No	No	Yes	No
I	Interest	$0\cdots1\cdots\infty$	Yes*	Yes	Yes	Yes	No	No	No	No
IS	Cosine	$0\cdots\sqrt{P(A,B)}\cdots1$	No	Yes	Yes	Yes	No	No	No	Yes
PS	Piatetsky-Shapiro's	$-0.25\cdots0\cdots0.25$	Yes	Yes	Yes	Yes	No	Yes	Yes	No
F	Certainty factor	$-1\cdots0\cdots1$	Yes	Yes	Yes	No**	No	No	Yes	No
AV	Added value	$-0.5\cdots0\cdots1$	Yes	Yes	Yes	No**	No	No	No	No
S	Collective strength	$0\cdots1\cdots\infty$	No	Yes	Yes	Yes	No	Yes*	Yes	No
ζ	Jaccard	$0\cdots1$	No	Yes	Yes	Yes	No	No	No	Yes

where: P1: $O(\mathbf{M}) = 0$ if $det(\mathbf{M}) = 0$, i.e., whenever A and B are statistically independent.
P2: $O(\mathbf{M_2}) > O(\mathbf{M_1})$ if $\mathbf{M_2} = \mathbf{M_1} + [k\ -k;\ -k\ k]$.
P3: $O(\mathbf{M_2}) < O(\mathbf{M_1})$ if $\mathbf{M_2} = \mathbf{M_1} + [0\ k;\ 0\ -k]$ or $\mathbf{M_2} = \mathbf{M_1} + [0\ 0;\ k\ -k]$.
O1: Property 1: Symmetry under variable permutation.
O2: Property 2: Row and Column scaling invariance.
O3: Property 3: Antisymmetry under row or column permutation.
O3': Property 4: Inversion invariance.
O4: Property 5: Null invariance.
Yes*: Yes if measure is normalized.
No*: Symmetry under row or column permutation.
No**: No unless the measure is symmetrized by taking $\max(M(A,B), M(B,A))$.

Figure 4.1: Interestingness measures and their properties. Source: [79] (abridged)

Perhaps the biggest advantage of *GUHA* is that it enables mining not only for the conventional confidence and support based association rules, but also for rules corresponding to various additional relations of Boolean attributes including relations described by statistical hypotheses tests. Arules package also offers various statistical hypotheses tests, but it restricts the use of most if the tests only on rule sorting and pruning. In addition to confidence and support, *extended rule selection measures* described in paragraph 4.2.2 can be used.

After reviewing some of the literature referenced by Arules documentation [43], we found the following Arules measures to coincide with *4ft-Miner* quantifiers:

- χ^2 **measure** is available in the *LISp-Miner* under the same name. χ^2 test is based on the comparison of observed frequencies with the corresponding expected frequencies. The less close are the observed frequencies to the expected frequencies, the greater is the weight of evidence in favor of dependence.

- **conviction** – we were not able to map it to a particular LM quantifier, but we found a note in [45] that it is shown in Hájek(2003) that the conviction quantifier is associational, which means that it falls within the logic of association rules as defined e.g. in [45].

- **oddsRatio** – As an alternative to χ^2 measure, LISP-Miner also offers the "exact" Fisher test, since χ^2 measure test is not suitable when the "expected values" in any of the four cells of the table are below 10 and there is only one degree of freedom. The null hypothesis for the Fisher test is that the Odds ratio equals 1. For Odds ratio greater than one, the value of Fisher test monotonically increases with the Odds ratio. Therefore, it could be assumed that sorting of the association rules according to Fisher test or Odds ratio should result into the same order.[8]

[8] In our experiments the value of Fisher test was alway 0.

- **Improvement** – the improvement of a rule is the minimum difference between its confidence and the confidence of any proper sub-rule with the same consequent. This feature offers the *LISp-Miner* procedure SD4ft-Miner[9].
- **Absolute difference of improvement to 1** - corresponds to the Above Average Implication quantifier which is defined as that "among objects satisfying LHS there must be at least 100*p [%] more objects satisfying RHS than there are objects satisfying RHS in the whole data matrix".

In the experiments which are presented in Chapter 7 we decided to use Support, Confidence, Above Average Quantifier (and its Arules equivalent) and a typical LM quantifier Double Founded Implication.

4.5 Clustering

Based on the review of the available literature [17, 7, 25] on clustering in WUM, the approach presented in [25] was chosen. This paper summarizes several years of work of a group of researchers from a renown institute – Palo Alto Research Center. Out of all the research papers reviewed, their work seemed to be the most suitable for e-commerce use. The input data needed are relatively easily obtainable, the main algorithm used is a variant of the most widely used K-means algorithm.

Although a special data representation is required, the preprocessing is straightforward and does not significantly differ from the visitor profiles as proposed in Chapter 3. The time complexity of K-means does not rise as dramatically with the dimensionality of data as it is with associations. Therefore, the researchers did not have to address the curse of dimensionality. The essence of the data preprocessing task used in [25] is presented in the case study Section 7.2.

All the algorithms presented here are Multi Modal Clustering (MMC). This approach allows to combine various modalities such as page topic or category during the clustering of the visits. Each modality is represented as a subvector, whose elements correspond to weights assigned to each of the original attributes[10]. The advantage of using MMC instead of regular clustering is that each modality may be assigned a different weight. These weights are different ones than the weights assigned to the individual components of the visitor profile on the basis of for example Score.

The similarity of the input vectors is computed using the cosine similarity measure on a per modality basis. The similarities of the modalities are linearly combined using the modality weights. The total similarity of two visitor profiles UP_i, UP_j is computed as follows:

$$d(UP_i, UP_j) = \sum_{m \in Modalities} w_m cos(UP_i^m, UP_j^m) \tag{4.1}$$

$$\sum_m w_m = 1 \tag{4.2}$$

Because cosine measure is used, the input vectors must be normalized first. Normalization is done on a per modality basis, each modality (subvector) is normalized to a unit length. This

[9] Task parameters: The second set is treated as a subset specification for the first set

[10] Some theoretical introduction for using modalities is given in the beginning of Chapter 3, Section 7.2 contains a practical example and more detailed information.

step is taken care of by the clustering algorithm; it is not needed to do it in the preprocessing stage.

The clustering algorithm used in [25] was recursive bisection K-means. This algorithm and its two variants are introduced here. The first baseline variant is Spherical K-means. The second variant is the author's attempt to improve the clustering performance of the Repeated Bisection K-means with a genetic algorithm.

4.6 Spherical K-means

Spherical K-means differs from the "classical" K-means in that it does not use Euclidian distance, but the cosine similarity measure to compute how instances resemble one another. Cosine similarity measure is widely used in information retrieval. It was proven in [76] (cited according to [88]) that it gives much better results for high dimensional data such as shopping baskets and textual documents then the Euclidean distance[11].

Algorithm: Spherical K-means

Input: Unit length vectors (instances) $\chi = x_1, \ldots, x_N$ in R^d and the number of the desired clusters K. All input vectors are of dimension t.

Output: Partitioning of the input vectors into clusters as specified by a membership vector $Y = y_1, \ldots, y_N, y_n \in 1, \ldots, K$

Steps:

1. Initialization: initialize the individual input unit-length vectors - the start centroids $\mu_1, \ldots, \mu_K$.
2. Membership assignment: to each vector x_n assign $y_n = argmax\ x_n^T \mu_K$
3. Recalculate centroids: for cluster k, let $X_k = \{x_n | y_n = k\}$. Centroid is computed using

$$\mu_k = \frac{\sum_{x \in x_k} x}{\left\| \sum_{x \in x_k} x \right\|} \tag{4.3}$$

4. Stopping condition: stop when y does not change, otherwise go to Step 2.

4.6.1 Initialization

Maximum attention should be devoted to the means of algorithm initialization, because it determines the result of the clustering to a large extent. The individual implementations of the K-means algorithm differ often in the way of algorithm initialization. The strategies most often used are Forgy, MacQueen, Kaufman, random. The winner of the comparison of these algorithms [63] is the Kaufman version.

[11] Interestingly, if both the document vector and the query vector are unit-length, then using the Euclidean distance leads to the same results as the cosine similarity measure, the proof is given in [88].

This comparison was not, however, carried out for the cosine similarity measure as a distance metric, but for the more commonly used Euclidean distance, therefore it may not fully apply to the approach taken here which is based on the initialization strategy proposed by Forgy in 1965. In this approach, K instances are chosen by random from the dataset and they become the K starting centroids. Each of the remaining instances is assigned to the centroid with which they have the largest similarity (in the original approach lowest distance). In this way K clusters are be formed.

Apart from the chosen initialization algorithm, the K number of desired clusters has also a high influence on the outcome. The number K is chosen Apriori.

4.6.2 Assigning Instances To Clusters

In Spherical K-means, the similarity of an instance with all the cluster centroids is computed and the instance is assigned to the cluster with whose centroid it has the largest similarity. For simplicity sake, modality weights are disregarded in the following formula.

$$k(x) = argmax_{k \in 1,\ldots,K} \frac{\sum_{s=0}^{t}(\mu_{ks} * x_s)}{\sqrt{\sum_{s=0}^{t}(\mu_{ks})^2 * \sum_{k=0}^{t}(x_s)^2}} \tag{4.4}$$

If both the centroid vector and the instance vector are normalized, the formula can be simplified [88]

$$k(x) = argmax_k x^T \mu_k \tag{4.5}$$

In "normal" k-meas, the cluster vector is assigned based on the minimal Euclidean distance

$$k(x) = argmin_{k \in 1,\ldots,K} \|x - \mu_k\| \tag{4.6}$$

For Spherical K-means, the direction of the vector is more important than the vector length. If one vector is a multiple of another vector, these vectors are considered as identical. As a consequence, two different instances may have zero distance from one another when the distance is computed with cosine measure. In the second step of the algorithm, in which cluster membership for instances is determined, one special situation has to be taken care of. There are cases, when the computed centroid-instance similarity is the same for multiple centroids. Because K-means does not create probabilistic clusters, there are the following two possible strategies:

1. Assign the instance to the cluster with the smallest number of instances
2. Choose one of the the clusters by random (with even probability)

The advantage of the first approach is that it lowers the probability of the emergence of empty clusters. This strategy may however lead to biased results. The implementation on the thesis CD, which is also used in the Chapter 7, uses the second strategy.

4.6.3 Recomputing The Centroids

During the phase of assigning instances to the clusters, the existing centroids gradually cease to correctly represent the clusters. It is therefore necessary to recalculate the centroids for each of the clusters. Cluster centroid is calculated as a mean of the instance vectors belonging to the cluster. In order to assure that it will be possible to use the simpler formula 4.5 for cosine measure calculation in Step 2, it is necessary to assure that the centroids are unit-length.

4.6.4 Stopping Condition

The K-means algorithm stops when there is no change in the assignment of instances to clusters during two consecutive algorithm iterations, or there is no change in the intra-cluster similarity in any of the clusters. An alternative stopping condition which is mentioned in the literature is halting the algorithm after certain maximum number of iterations is reached. However, the K-means algorithm is very popular because it very quickly converges[12]; this stopping condition was not therefor in the Chapter 7 implementation utilized.

4.6.5 K-means Disadvantages

K-means is a simple and powerful algorithm. There is a price to pay for the simplicity, the following are its commonly acknowledged disadvantages:

- Generally not suitable for situations where overlapping clusters are natural
- Presence of outliers has a highly unfavourable impact on the outcome of the clustering
- Membership of instance to a cluster is specified by a true/false value, fuzzy membership values are not supported
- Finding a global optimum is not guaranteed

The last disadvantage is perhaps the most serious one. The outcome of the clustering is dependent on the choice of the initial set of centroids and on the number of desired clusters. Witten & Frank [86] illustrate this disadvantage with the following example

> Consider four instances arranged at the vertices of a rectangle in twodimensional space. There are two natural clusters, formed by grouping together the two vertices at either end of a short side. But suppose that the two initial cluster centers happen to fall at the midpoints of the long sides. This forms a stable configuration. The two clusters each contain the two instances at either end of a long sideno matter how great the difference between the long and the short sides.

It is therefore recommended to run the entire clustering algorithm several times with different parameters and choose the best results (with the lowest total distance).

4.7 Repeated Bisection K-means

Repeated Bisection K-means is a variant of the K-means algorithm. This algorithm starts with one cluster which contains all the instances. This cluster is bisected with standard K-means into two clusters. One of the clusters is based on some criterion (largest cluster, cluster with the smallest total similarity) selected and bisected – replaced by two new clusters. One of the clusters is again chosen and bisected. This cycle continues until the K number of clusters is obtained [60].

Algorithm: Repeated Bisection K-means

[12]The high speed of the K-means algorithm is based on empirical results. Determining the number of iterations in which the algorithm converges *analytically* is an open problem [13].

Input: A set of unit-length vectors $\chi = x_1, \ldots, x_N$ in R^d and the number of output clusters K.

Output: Partitioning of data vectors into clusters given by the membership vector $Y = y_1, \ldots, y_N, y_n \in 1, \ldots, K$

Steps:

1. Initialization: Place all instance vectors into one cluster
2. Selection step: Choose the cluster C to be bisected
3. Bisecting step: Use K-means to partition the cluster C into two sub-clusters
4. Repeating step: Repeat Step 2 ITER-times and choose the partitioning with the highest overall similarity
5. Stopping condition: Stop when the desired number of clusters is achieved, otherwise go to Step 2

Implementation note In the program present on the thesis CD, the largest cluster was selected (Step 2). Setting the number ITER was left on the user.

Number of iterations of the Bisecting step has a large influence on the time complexity of the algorithm. Each iteration of the Bisecting step triggers an execution of the K-means algorithm. For K clusters and $ITER$ iterations of Step 3 during each partitioning, the total number of iterations P_{rep} of the Step 3 is given by the following formula

$$P_{rep} = ITER * (K - 1) \tag{4.7}$$

Repeated Bisection K-means give, according to the researchers from Palo Alto Reserach Center [25], the best results as compared to a range of other clustering algorithms which they also tried. On the other hand, these researchers were not able to find a reliable method, which would be able to determine the optimal number of clusters K.

4.8 Repeated Bisection K-means With Genetic Algorithm Initialization

One of the main disadvantages of the K-means algorithm is, apart from the necessity to set the number of clusters beforehand, its high sensitivity to the selection of the K starting centroids. The attempt to overcome this problem presented here and proposed and implemented by the thesis author uses a genetic algorithm to initialize the individual runs of K-means in the Bisecting step of Repeated Bisection K-means. This section uses some notions from the theory of evolutionary computation whose explanation is beyond the scope of this thesis. Please refer e.g. to [14] when needed.

Algorithm: Repeated Bisection K-means With Genetic Algorithm Initialization

Steps:

1. Initialization: Place all instance vectors into one cluster
2. Selection step: Choose the cluster C to be bisected.
3. Bisecting step: Use K-means initialized by the genetic algorithm to partition the cluster C into two sub-clusters
4. Stopping condition: Stop when the desired number of clusters is achieved, otherwise go to Step 2

It can be seen that this algorithm differs in a way how it copes with the "volatility" of the classical K-means. Instead of blindly repeating standard K-means with different randomly chosen vectors from the instance space assigned as the starting centroids, a genetic algorithms is used. The specific algorithm used is an evolutionary algorithm with tournament selection. Starting vectors belonging to the best partitionings achieved are evolved using mutation and crossover in order to find such starting centroids that will lead to the best clustering result and the highest total similarity.

Evolutionary algorithms do not work with instances or vectors but with *individuals*. What concretly is an individual depends on the problem solved; it should represent a possible solution. The individual here is composed by two starting centroid vectors needed for the initialization of K-means in the bisecting step. Thanks to the use of crossover and mutation operators, the two centroids will not be restricted to contain only instance vectors, which actually exist in the clustered data. This of course applies to second and further generations, the first generation is assigned vectors from the input instance space.

Algorithm: Bisecting step with the genetic algorithm initialization

Input: Cluster to be bisected represented by unit-length vectors $\chi = x_1, \ldots, x_N$ v R^d.

Output: Partitioning of the input vectors into two clusters

1. For each of the elements $x_1, \ldots, x_N$ find the maximum value of the component max_n and minimum min_n. These values will be used for mutation during the tournament selection
2. Initialize population P of $s * v$ individuals $J \in \chi$, where s is the number of groups and v the group size
3. Assign two random vectors from χ to every individual
4. Partition P into groups of s individuals.
5. Carry out tournament selection
6. Stop when the maximum number of epochs E was reached or when fitness have not changed during $Stag$ past epochs. Otherwise goto Step 4.

Implementation note Tournament selection was programmed for the standard group size, which is 4 according to [14] page 32. The advantage of this algorithm is that it guarantees the survival of the best individuals. Maximum fitness reached can not therefore drop as the evolution proceeds. In practice, it can sometimes happen that just before convergence occurs and the algorithm is stopped, the fitness in the iteration $i + 1$ slightly (in the range of tens of thousandths of percent) drops below the value of fitness of the iteration i. This effect is caused by a rounding error which happens even with double precision variables.

The tournament selection also prevents a premature loss of diversity in the population. Should an individual have a much higher fitness than all other members of the population, she can still have only one descendant.

Algorithm: Simple Tournament Selection

Input: Group of n individuals of generation i

Output: Group of n individuals of generation $(i + 1)$

1. Evaluate the individuals and choose $\frac{n}{2}$ best ones a $\frac{n}{2}$ worst ones.
2. Create descendants by copying the best individuals
3. Cross descendants with the probability of 50%
4. Mutate each individual with the probability of 50%
5. Replace the worst individuals by the descendants

Evaluation of an individual is the total similarity returned by the run of the K-means algorithm with the starting vectors specified by the individual. If an individual does not undergo any change during two consecutive epochs, because it was neither chosen for mutation nor for crossover, the individual is not evaluated.

When the probability of mutation is $P(mut) = 50\%$ and $P(cros) = 50\%$, this situation occurs with probability of 25% for descendants, because mutation and crossover are independent events. The overall probability $P(eval)$ that the fitness value will be recomputed for an individual is only 37.5% for the second and consecutive epochs. The reason is that 50% of individuals in each epoch are parents from the previous epoch for which the probability of change due to mutation is zero and therefore their fitness values can be more often "recycled".

$$P(eval|epoch > 1) = 0,5 + 0,5 * (P(mut) + (1 - P(mut)) * P(cros)) \tag{4.8}$$

$$P(eval|epoch = 1) = 1 \tag{4.9}$$

4.8.1 Crossover

The crossover of individuals is done separately for each of the modalities, though the probability whether the individuals will be selected for crossover is given once for all modalities. The crossover positions are chosen for each of the modalities randomly. The components of each of the instance vectors are left unchanged up to the crossover position, from the crossover position onwards, the components are exchanged between the two vectors as illustrated on

Modality 1				Modality 2					
12	20	45	24	26	43	21	64	26	16
35	83	46	75	35	94	31	84	23	35

Table 4.3: Two centroid vectors before crossover

Modality 1				Modality 2					
12	20	*46*	*75*	26	43	21	*84*	*23*	*35*
35	83	*45*	*24*	35	94	31	*64*	*26*	*16*

Table 4.4: Two instance vectors after crossover, the crossover position for the first modality was 2, for second modality 3

Tables 4.8.1. These tables depict the crossover of only one of the two vectors contained in an individual.

There are numerous heuristics for enhancing the crossover operator. It is a widely held belief that coming up with such a heuristic can deteriorate the performance of the genetic algorithm rather than enhance it due to unexpected side effects, which undermine the balance set by the mutation and crossover operators. Despite this warning, one heuristic was introduced.

Each individual represents two centroids. For crossover, it is necessary to choose the couples who should exchange the genetic information with one another. This decision can be made by random, or it is possible to choose the couples in a way which maximizes their total similarity. This heuristic is hoped to increase the positive effects of crossover. It was used during the experiments presented in Chapter 7, but this option can be turned off.

4.8.2 Mutation

Mutation is Gaussian. An individual is mutated with 50% probability. If the individual is selected for mutation, all components of both centroid vectors which the individual represents are subject to mutation. Each vector component n is mutated separately with the use of maximum max_n a minimum min_n of this component[13]. These values are then used for computation of the standard deviation for Gaussian mutation

$$\sigma = \frac{max_n - min_n}{SigFac} \tag{4.10}$$

$SigFac$ parameter sets what portion of the interval denoted by the minimal value min_n and maximal value max_n should be used as the standard deviation of the mutation. The mean value of the Gaussian mutation is always zero. Each of the components $x_1...x_n$ of the mutated instance vector is changed according to the following formula

$$x_i = x_i + N(0, \sigma) \tag{4.11}$$

Implementation note Random values from the normal distribution are generated using free component written by Stefan Troschuetze, which is avaiable at `http://www.troschuetz.de/Random.aspx`.

[13]These values are precomputed when the algorithm "Bisecting step with the genetic algorithm initialization" is run.

4.9 Choosing The Right Data Mining Solution

According to the CRISP-DM Data Mining Guide 1.0 [24] the actual *data modeling phase*, when the data-mining algorithms are applied, takes usually only about 10-20% of the resources and effort needed to deploy a data mining solution. Nevertheless, this chapter showed that learning the intricacies and terminology of a specific existing data mining solution is often no easy task. Programming a one-purpose data-mining application that is built upon some widely-described algorithm is a serious option. There are however some pitfalls.

First, if the program is not thoroughly tested, it may easily deliver flawed results. It may be too late to find out that it omitted obvious patterns on one hand, or made up not existing ones on the other. Therefore, in a production environment it still highly desirable to compare the results of the newly designed program with the results returned by an existing solution.

Secondly, when a new data mining program is designed, one is tempted to fix some of the "obvious shortcomings" of the data-mining algorithms implemented. It may easily happen that a lot of effort will be exerted into a complicated solution without any reward in the improvement of the mining results. This was actually the case with the Repeated Bisection K-means with Genetic Algorithm initialization presented here and demonstrated in Chapter 7.

Selecting the most suitable data-mining algorithm for a given task is not an easy task as well. Chapter 7 shows a typical use of clustering and association rule mining on web usage data. The ability to choose the right algorithm for a given task comes naturally with experience, there are, however, more exact approaches to aid this process.

For example [87] proposes WALMAGE, an evaluation framework which recommends the most suitable mining algorithm for a particular E-commerce scenario based on cognitive nature of the browsing patterns exhibited by the website's visitors as well as the structure of the website. This algorithm basically represents each of the considered algorithms as a vector, whose components denote how good a particular algorithm is in a given field. An example component is for example *backtracking tendency*. A high value of this component for a particular algorithm means that it can cope well with backtracking, low value means the opposite. A vector with the same meaning of components is used to describe the E-commerce scenario – the website and its visitors. Then the Euclidean distance between the scenario and each of the algorithms is computed. The algorithm with the lowest distance wins.

There are still some open theoretical questions pertaining to the WALMAGE algorithm acknowledged by the authors. In addition, the vectors representing each of the algorithms are set arbitrarily. It can be concluded that although exact approaches to choosing data-mining algorithms exist, it is still the data analyst who shall have the last word.

Part II

Case Study

Case Study Foreword

The practical part of this thesis complements the theoretical part in that it provides an in-depth coverage of selected concepts in real world situations.

Chapter 5 shows how easy it is to carry out conversion fraud presented in Chapter 1 in practice on a widely used clickstream analysis solution.

Chapter 6 presents *GAIN*, a data collection software written by the thesis author, which is an answer to the Google Analytics vulnerability presented in Chapter 5. In addition, this software has the capabilities to gather information needed to create the training set required by the Classification-based approach to dimensionality reduction proposed in Section 3.7. *GAIN* uses the recommended approach to primary data collection suggested in Section 2.1 as well as it includes some abilities for acquisition of meta-data, which are the ground stone of modern Web Usage Mining as explained in Section 2.2.

Chapter 7 presents a case study use of the data mining algorithms presented in Chapter 4. Section 7.2 gives an example of how the visitor profiles introduced in Chapter 3 could be built in a particular WUM scenario.

Chapter 5

Conversion Fraud in Google Analytics - A Proof of Concept Attack

Google Analytics is a widely recognized tool, which helps web masters all over the world to analyze traffic on their website. This software has quickly become a benchmark for other click stream analysis software. This chapter presents this software in a less favourable light pointing at its shortcomings to prevent and detect click- and conversion fraud[1].

This chapter is organized as follows. First, an overview of Google Analytics and its relation to on-line advertising is given. Section 5.1 details how Google Analytics collects data. This section is largely based on the author's own inspection of the Google Analytics tracking module. This provides a foundation for the description of attack in Section 5.2, which is first explained generally and than shown on a real website. The chapter is concluded with Section 5.3, which describes the proposed defense against the attack.

Google Analytics (GA) is a free service, which collects web usage data and provides web interface to statistics. First, we will try to discuss the reasons that lie behind the popularity Google Analytics (GA) has gained in a short time since its launch in November 2005. There were many free traffic analysis services on the Internet before 2005. Some of them also provided many of the statistics Google Analytics provides. Its main differentiator can be summed up by the following quotation from the news article from the November 2005 [58]:

> [Google Analytics] lets companies see exactly how visitors interact with their Web site and how advertising campaigns are faring.

What differs Google Analyics from most other free services is powerful campaign and conversion tracking. These features are supposed not only to help existing users of Google's advertising programs AdSense and AdWords, but they are also intended to encourage the users of Google Analytics using other (or none) advertising schemes to start using Google AdSense/AdWords. That is achieved among other means by building trust in that the effectiveness of the ads can be easily and precisely measured by Google Analytics. Google Analytics can provide report on conversion rate of visitors by

[1] For example, security is given as the number 3 reason why not to adopt Google Analytics by its competitor [6].

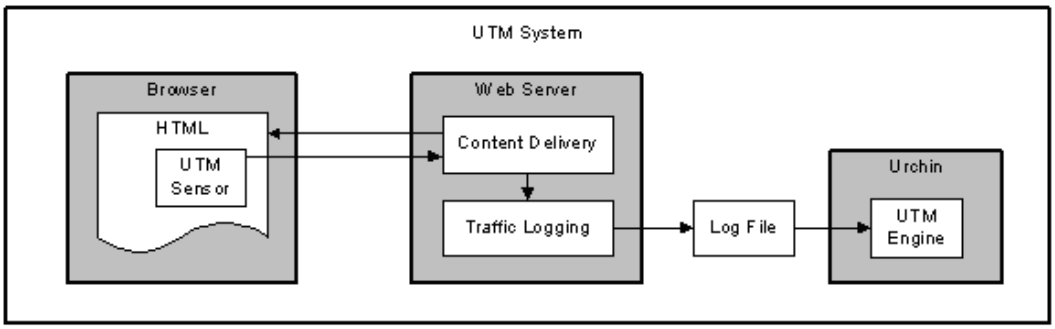


Figure 5.1: UTM System Architecture. Source:`http://www.google.com/support/urchin45/bin/answer.py?answer=28710`

- CPC Advertising Campaigns
- Referrals incl. affiliate partners
- Organic search by individual search engines
- Direct (no referral)
- Keywords

The users of Google Analytics are not currently recommended to cross-validate the statistics on conversions and visits provided by Google Analytics by other service or software. As we will show later, the current way of collecting data by Google Analytics leaves space for fraud. If a cross validation using server log files were utilized in addition to reports provided by Google Analytics, the possibilities of fraud would be limited. Unfortunately, as we will see later, a page view recorded by Google Analytics may not result into a page view recorded in a server log of the tracked site.

5.1 Data Collection In Google Analytics

Google Analytics uses javascript to track visitors. The webmaster has to include a small snippet of code on each of the web pages that she wants to track. This code tracks page views and user actions and sends this information to Google for further processing. This technique is quite common among web analytics software. What is interesting is how the data are collected on the client side.

The architecture of UTM Sensor is depicted on Figure 5.1.

It should be noted that some parts of Google Analytics are protected by US. Patent 6804701 from October 12, 2004 assigned to Urchin Software Corporation. Urchin Software Corporation was acquired by Google in 2005 prior to release of Google Analytics.

5.1.1 UTM Sensor

UTM Sensor is about 500 lines long javascript code `urchin.js`. The code has size approximately 21 KB. For comparison, the client tracking code `guj.js` used by `clicktracks.com`, a higher tier paid service used e.g. by Nokia, Coca-Cola or University of Cambridge, is only

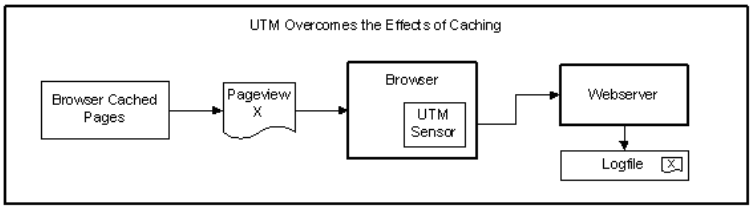


Figure 5.2: UTM System prevents biases caused by caching. Source:`http://www.google.com/support/urchin45/bin/answer.py?answer=28710`

475 lines long and has only 14 KB and that despite the fact that the latter contains comments and descriptive variable names, while `urchin.js` contains very short variable names, and very short aliases for various properties of the widely used `document` object and almost no comments. For example, variable `_udl` is used to refer to `document.location`. As a consequence, the code `urchin.js` is relatively long and difficult to read.

When a page is requested by a visitor, during the page load the visitor's browser executes `urchin.js` script[2]. This script is either present in the browser cache or downloaded from `http://www.google-analytics.com`. The script collects the data for a particular page view and then requests a 1×1 image `__utm.gif` from `http://www.google-analytics.com`. The request URI contains a querystring which encapsulates all the collected information from the client including the visitor id, visit sequence number, referral and visited page. The data are sent in a proprietary publicly undocumented format.

UTM Sensor offers, with some limitations, the following two important features:

1. It fights caching (Figure 2) and proxying (Figure 5.1.1) by forcing an image to be downloaded by the browser and by a patent pending combination of browser cookies (see figures). However, this mechanism does not ensure that the tracked site receives a hit as the `__utm.gif` is downloaded from `http://www.google-analytics.com`.

2. It uniquely identifies each visitor by first party cookies. According to web analytics software vendor [4], first-party cookies are not very likely to be deleted by the user. However more independent surveys [27] conclude that "Third party cookies are deleted at nearly same rate as first-party cookies."

The vulnerability of Google Analytics described further stems from the way how UTM Sensor hands over the collected data to Google for further processing. This process is one way – the information is passed from the visitor's browser to the part of the Urchin system hosted at Google without any acknowledgment sent to the tracked site.

5.1.2 Forging Conversions In Google Analytics

The operation of Urchin Sensor is fully transparent both during the transmission of the data and in the way they are created. Any javascript debugger can be used to intercept the data `urchin.js` sends. An attacker can derive from the Urchin Sensor code stored in `urchin.js` the knowledge how to create valid querystrings containing the desired (spoofed) information. This information is sent to Google Analytics by requesting the `__utm.gif` file and including the

[2]The script can be called also after a page load in order to send information about a user action (such as expanding a hidden part of a page) to Google Analytics.

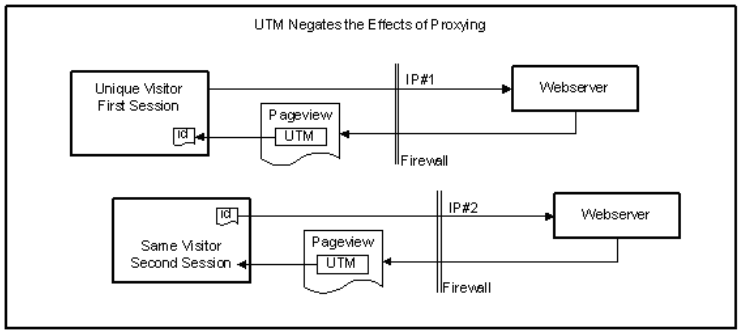


Figure 5.3: UTM System prevents biases caused by proxying. Source:`http://www.google.com/support/urchin45/bin/answer.py?answer=28710`

spoofed information into the querystring. By doing so, the attacker can spoof any sequence of pageviews or actions and forge the referrer. The attacker is not limited to visits of length one as with the modified attack introduced by Anupam et al (see Section 1.1.1). Interestingly, the attacker does not need to send any requests to the tracked site, whose visits and conversions are being forged.

In the next subsection we will discuss the format in which Urchin Sensor sends information. This information may not be necessarily completely accurate as the `urchin.js` code is difficult to read and there is little means of knowing the purpose of several fields. However, our understanding of this format was sufficient to reconstruct the user sessions and to successfully carry out a proof-of-concept attack on one site tracked by Google Analytics.

5.1.3 Exchange Format

The information sent in the querystring of the request for `__utm.gif` is largely a marshalling of values kept in browser cookies. Therefore, we will first explain the cookie Urchin Sensor sets to the visitor browser and then we will proceed to the exhange format.

When the urchin tracker detects a new visitor, it sets four first-party cookies - `_utma`, `_utmb`, `_utmc`, `_utmz`. These cookies are set to expire after a different time period. `_utma` is a permanent cookie, its expiration is always set to "Sun, 18 Jan 2038 00:00:00 GMT". This cookie carries most of the information. It comprises of six numbers delimited by dots "X1.X2.X3.X4.X5.N". An example value of _utma cookie is
96992031.169673154.1171745298.1171794262.1171795994.4.

The individual components of th `_utma` cookie are

X1 contains by default a hash of the domain name of the tracked site. The hash is computed in real time in the `urchin.js`. If the tracked site uses a check out located on a third party web site and wants to continue to track the visitor when she proceeds to the check out, the `_uhash` variable must be set to *off* instructing the script to set the value of X1 always to *1*.

X2 Random number between 0 and 2147483641 generated during the first visit. This is in fact a visitor ID on the tracked site. For most sites, the probability of two visitors obtaining the same number is very small. This range alone might be insufficient for world's most

frequently visited web sites. The probability that a new visitor will be assigned the same ID as an existing visitor for a website with two million visitors is approximately 1:1000. However, the X3, X4, X5 and N elements provide enough information to detect this in most cases.

X3 Time stamp of the first visit. It is a (rounded) number of milliseconds from midnight of January 1, 1970 divided by number 1000. Server time is used. This number is generated at the first page view of the visit.

X4 Time stamp of a visit with sequence number $N - 1$, where N is a sequence number of the current visit. During the first and second visit, it contains the same value as X3.

X5 Time stamp of a visit with sequence number N. During the first visit, it contains the same value as X3. This number is generated at the first page view of the most recent (current) visit.

N Visit sequence number. First visit has sequence number 1. Second and further visits are detected, when `__utma` cookie is found and either cookie `__utmb` or `__utmc` is not found. When a new visit of a returning visitor is detected, N is incremented, X3 shifts to X4, X4 to X5 and X3 is computed based on current time on the *visitor's* computer.

`__utmb` is a cookie, which is set to expire after number of seconds set in the `__utimeout` variable (might be changed by the web master). This cookie is used to give the upper bound to a maximum time interval between two requests within a session. This cookie is not usually deleted (depends on the browser and its setting) when the user leaves the site or shuts down the browser. In the trivial case, `__utmb` contains only the domain hash or value "1" (the same value as X1 of `__utma`).

`__utmc` is a session cookie - it expires at the end of the session. The definition of when the session ends might be dependent on the browser. While `__utmb` is not typically deleted, when the browser shuts down, most browsers delete `__utmc`. In the trivial case, `__utmc` contains only the domain hash or value "1" (the same value as X1 of `__utma`).

`__utmz` is a cookie used to track referral information accross visits. It comprises of five elements delimited by dots in the form of Y1.Y2.Y3.Y4.R. First four elements are numeric, the last one is a string value, which can be further divided into several fields. An example values for two consecutive visits are

- `1.1178635257.1.1.utmccn=(direct)|utmcsr=(direct)|utmcmd=(none)`
- `1.1178962390.2.2.utmccn=(organic)|utmcsr=google|`
  `utmctr=poznani|utmcmd=organic`

The visitor first came to the website by directly entering its address. When she wanted to visit the website for the second time, she could not recall the website URL address. She looked it up on organic search engine Google using the term "poznani" and entered the website from Google's SERP.

Y1 This value contains the domain hash or value "1". It is computed in the same way as X1 of `__utma`.

Y2 Timestamp of the last visit when R changed. It is computed in the same way as X5 of `_utma`.

Y3 A sequence number of the visit to which the values stored in R element belong.

Y4 Number of times the visitor came to the site with a different referrer than in a previous visit. Also the number of times the R element changed.

R Information about the visitor's referral. It comprises of several elements including: `utmccn`, `utmcsr`, `utmcmd` and `utmctr` delimited by a pipe sign "|". `Utmccn` is an abbreviation for campaign name, `utmcsr` stands for referral medium, `utmcsr` for referral source, `utmctr` is term/keyword. Urchin Sensor tries to extract these values from the referrer querystring[3]. If they are not present, they are assigned based on the logic explained below.

The value of R is not overwritten with a direct visit to the website. However, if the first visit to the website is direct i.e. with no referral, the R field is set to "utmccn=(direct)|utmcsr=(direct)|utmcmd=(none)" and the rest of the fields are set accordingly. By default R (and consequently all other elements except Y1) are updated, when the visitor starts a new session by following some link from a different website – be it a PPC advertising, organic search or generic referral link. This behaviour can be changed if the referring link's querystring contains `utm_nooverride` parameter set to 1, which ensures that this referral will not overwrite existing value stored in R.

If the referring domain is found to correspond with the $i-th$ element of the `_uOsr` array[4], `Utmccn` is set to "(organic)", `utmcmd` to "organic" and `utmcsr` to `_uOsr[i]`. Next, Urchin Sensor tries to extract the sought term by looking for the value of a parameter named `_uOkw[i]` in the referrer and stores it in `_utmctr`.

If the referring domain is not found in the `_uOsr` list, then `Utmccn` is set to "(referral)", `utmcsr` is set to the referring domain, `utmcmd` is set to "referral" and `utmcct` is set to the relative part of path from the referrer.

The values stored in client side cookies constitute a large part of tracking information sent to Google. The remaining pieces of information are in most cases intuitive, therefore we include only a brief explanation:

- **utmwv** always equals 1. This attribute denotes the version of Urchin Sensor. Current version of Google Analytics (as of 18 June) uses version 1.0 (which has some enhancements compared to older Urchin Tracking Module 6.1.

- **utmn** is a unique id generated for each page view. For first pageview of a new visitor it is equal to X2 of _utma.

- A set of variables describing the page, browser and platform: **utmcs** – page charset, **utmsr** – screen resolution, **utmsc** – screen color depth, **utmul** – browser language, **utmje** – 1 if java enabled, **utmfl** –flash version

- **utmcn** set to 1 indicates a start of a new session or an update of utmz.

[3]If they are stored in parameters with different names, Urchin Sensor can be set to use these instead of the default ones.

[4]`_uOsr` List can be modified/extended by the webmaster to include local search engines.

- A group of variables describing the visited page: **utmhn** – host name, **utmp** relative url or action, **utmdt**– page title
- **utmr** full referrer as obtained from `document.referrer`.
- **utmac** – ID of the Google Analytics account including the ID of the tracked website within the account. Assigned by Google.
- **utmcc** – values stored in the utm cookies.

All the values are url encoded.

5.2 Attack Description

The possible attack could be carried out by requesting `__utm.gif` with appropriate parameters.

The attacker can fraud visiting pages and undertaking actions on them by constructing requests for `__utm.gif` containing faked information. To fraud one conversion, minimum three requests must be issued. In order to keep the conversion rate in not-suspicious bounds of e.g. 5%, the attacker could simulate additional 19 bogus visits per one conversion. Of course, these additional visits will not meet the conversion criteria. The whole sequence should be thoroughly randomized – visits should be of different paths, length and duration.

Consider the following three ways, how the forged request might be sent:

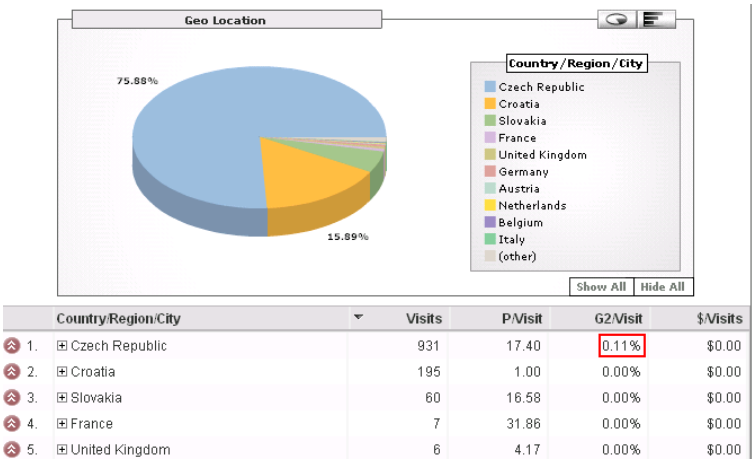
1. `Scenario 1` The attacker sends the requests from **her own computer**. In this case, the Referer HTTP request field can be easily spoofed to contain the url of site P. However, the requests are sent from the attackers IP address.
2. `Scenario 2` The attacker puts a client side script on site S, which instructs the browser to make forged requests. The script is executed by browsers of visitors of site S. Requests are made from undiscredited source IP addresses, but the referrer field contains url of site S not of site P.
3. `Scenario 3` The attacker can forge the source IP address and has access to fields contained in the HTTP header.

If all the requests originated from one IP as in the first case, a skilled user of Google Analysis could become quickly suspicious as she could observe from the Google Analytics reports that many conversions come from the same geographic location (see Figure 5.2 for an example report).

Therefore, reasonable attacks would be carried out by scenarios 2 or 3.

In order to disclose this kind of fraud, the advertiser would need to compare the data from Google Analytics with server web logs. Even when Google Analytics would report pageviews not present in the web logs, it might be difficult to prove to a third party that fraud occurred as it could be argued that this disproportion occurred due to caching. UTM Sensor by default does not assure that the page view will reach the original site and be recorded in the web server log. It makes sure only that `http://www.google-analytics.com` receievs a hit.

The last scenario is the most difficult to carry out, but it would be probably most difficult to detect as well. Attacks with forged source IP address have generally the disadvantage that



	Country/Region/City	Visits	P/Visit	G2/Visit	$/Visits
1.	Czech Republic	931	17.40	0.11%	$0.00
2.	Croatia	195	1.00	0.00%	$0.00
3.	Slovakia	60	16.58	0.00%	$0.00
4.	France	7	31.86	0.00%	$0.00
5.	United Kingdom	6	4.17	0.00%	$0.00

Figure 5.4: Geo Location Report Screenshot

the fraudster will not receive any acknowledgment or data sent in reply to her request. The recipient sends the data to the source IP of the fraudulent request, which is forged. The packets will therefore in the general scenario will not arrive back to the fraudster. Google Analytics does not send back any acknowledgment number that would need to be confirmed by a consecutive request. If it did, the possibilities for this dangerous kind of fraud would be severely limited. The principle of the proposal [12] of how to counter the CPC variant of this attack, would also be applicable in fighting this kind of fraud.

It is relatively difficult to forge an IP address, which is required by Scenario 3. However, if all the forged requests came from the same IP address as in Scenario 1, such an attack would be relatively easy to detect. The Scenario 2 could be thus probably the most appealing one to a prospective fraudster.

5.2.1 Proof Of Concept Attack

We will show, how to carry out a conversion inflation attack against website example.com. The attack is carried out by `frequentlyvisitedsite.com` on behalf of `cheatingpublisher.com`. The goal of the attack is to make the advertiser – the owner of example.com believe that visitors coming from `cheatingpublisher.com` met the conversion requirements and that the `cheatingpublisher.com` is eligible to receive a payment/commission for these visitors.

The malicious script `mal_ur.js` could be a modified `urchin.js` script located on frequently visited site. There are only a few changes, which need to be make in the script. Conceptually, the script should be changed not to use host name "frequentlyvisitedsite.com" but "example.com". In addition, instead of using the current name of the page on the frequently visited site, it would take a valid page or action name from `example.com` ensuring that this page/or action is reachable from the previous faked page and that conversions occur with moderate frequency. In this manner, all or only a sample of the traffic of the frequently visited site could become also (a faked) traffic of `example.com`.

If the `frequentlyvisitedsite.com` was not in an obvious way affiliated neither with `cheatingpublisher.com` nor with `example.com`, then even a close inspection of cheating publisher's website and its affiliates would not help the example.com to discover the malicious script. If Google Analytics provided a way to filter visits based on the Referer field from the

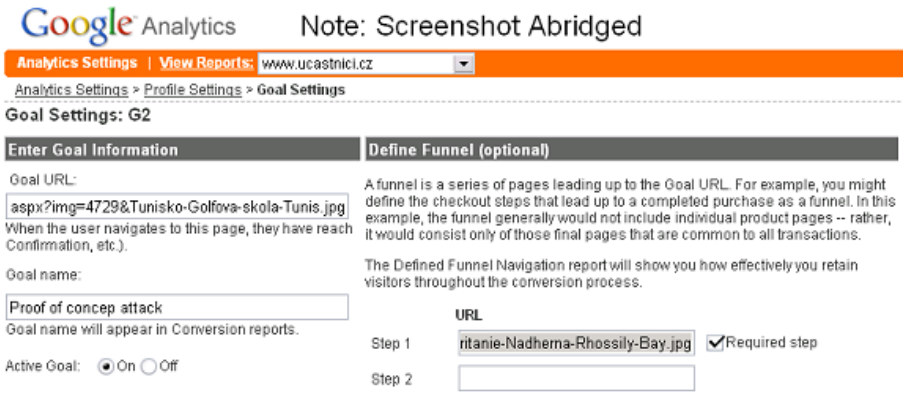


Figure 5.5: Google Analytics Conversion Goal Setting

HTTP header of requests for `__utm.gif`, then `frequentlyvisitedsite.com` could be easily disclosed.

The attacker could prevent this by serving pages containing an instruction to execute the `mal_ur.js` only to visitors with referrers turned off. In that case, it would be very difficult for example.com to detect fraud. It would have to notice a rise in the percentage of visitors with referrers turned off. The task of distinguishing the hits and conversions made by real users with turned off referrers from faked hits could be carried out by using additional information from the example.com server HTTP log file.

5.2.2 Example Attack

A visitor has to view the following pages in order to be converted: the `Landing page`, `Condion page 1` an `Condidition page 2`.

In our attack, the *Landing page* was `http://www.ucastnici.cz`, the URI of *Condition page 1* was /obrazek.aspx?img=1927&Velka-Britanie-Nadherna-Rhossily-Bay.jpg and the URI of *Condition page 2* was /obrazek.aspx?img=4729&Tunisko-Golfova-skola-Tunis.jpg.

Goals are used to track actions/conversions in Google Analytics. Therefore, we have set up the goal accordingly as seen on Figure 5.5. The attack comprises of three faked HTTP requests sent to google-analytics.com. No requests were, however, made for the *Landing page* and *Condition page 1*, *Condition page 2*.

```
=====REQUEST1=======

http://www.google-analytics.com/__utm.gif?utmwv=1&utmn=1727988091&utmcs=UTF-8
 &utmsr=1024x768&utmsc=32-bit&utmul=en-us&utmje=1&utmfl=9.0%20%20r45&utmcn=1
 &utmdt=%C3%9A%C4%8CASTN%C3%8DCI%20z%C3%A1jezdu%20-%20foto%20ze%20z%C3%A1jezd
 %C5%AF%2C%20fotogalerie%2C%20v%C3%ADce%20ne%C5%BE%208000%20fotek%20online
 &utmhn=www.ucastnici.cz&utmr=http://www.google.com/search?q=ucastnici&ie=
 utf-8 &oe=utf-8&aq=t&rls=org.mozilla:en-US:official&client=firefox-a&utmp=/
 &utmac=UA-135959-1&utmcc=__utma%3D105082119.1727988091.1178635495.1178635495
 .1178635495.1%3B%2B__utmb%3D105082119%3B%2B__utmc%3D105082119%3B%2B
 __utmz%3D105082119.1178635495.1.1.utmccn%3D(organic)
 %7Cutmcsr%3Dgoogle%7Cutmctr%3Ducastnici%7Cutmcmd%3Dorganic%3B%2B
```

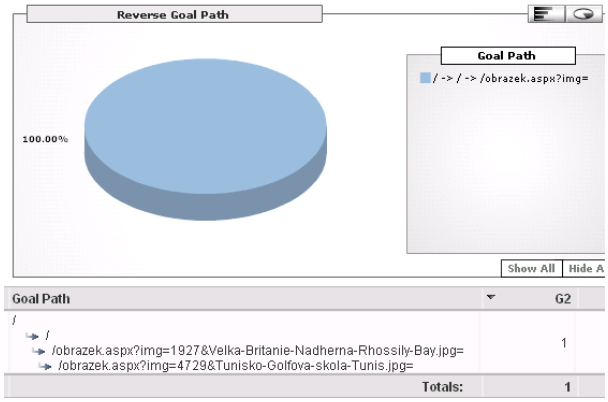


Figure 5.6: Reverse Goal Path In Google Analytics

```
=====REQUEST 2=======

http://www.google-analytics.com/__utm.gif?utmwv=1&utmn=1562324709&utmcs=UTF
 -8&utmsr=1024x768&utmsc=32-bit&utmul=en-us&utmje=1&utmfl=9.0%20%20r45&utmd
 t=%C3%9ACASTN%C3%8DCI.CZ%20-%20fotogalerie%20%C3%BAcastn%C3%ADku%20z%C3%
 A1jezdu& utmhn=www.ucastnici.cz&utmr=-&utmp=/obrazek.aspx?img=1927&
 Velka-Britanie-Nadherna-Rhossily-Bay.jpg&utmac=UA-135959-1
 &utmcc=__utma%3D105082119.1727988091.1178635495.1178635495.1178635495.1
 %3B%2B__utmb%3D105082119%3B%2B__utmc%3D105082119%3B%2B
 __utmz%3D105082119.1178635495.1.1.utmccn%3D(organic)
 %7Cutmcsr%3Dgoogle%7Cutmctr%3Ducastnici%7Cutmcmd%3Dorganic%3B%2B

=====REQUEST 3========

http://www.google-analytics.com/__utm.gif?utmwv=1&utmn=569397066
 &utmcs=UTF-8&utmsr=1024x768&utmsc=32-bit&utmul=en-us&utmje=1
 &utmfl=9.0%20%20r45&utmdt=%C3%9ACASTN%C3%8DCI.CZ%20-%20fotogalerie
 %20%C3%BAcastn%C3%ADku%20z%C3%A1jezdu&utmhn=www.ucastnici.cz
 &utmr=-&utmp=/obrazek.aspx?img=4729&Tunisko-Golfova-skola-Tunis.jpg
 &utmac=UA-135959-1&utmcc=__utma%3D105082119.1727988091.1178635495.
 1178635495.1178635495.1%3B%2B__utmb%3D105082119%3B%2B__utmc
 %3D105082119%3B%2B__utmz%3D105082119.1178635495.1.1.utmccn%3D(organic)
 %7Cutmcsr%3Dgoogle%7Cutmctr%3Ducastnici%7Cutmcmd%3Dorganic%3B%2B
```

Despite the fact that the referrers indicated in `utmr` and `utmz` part of `utmcc` do not match the HTTP referrer field of these requests, this attack succeeded and this fake visit appeared in the Google Analytics reports as if the the action/conversion requirement was met.

5.3 Proposed Defense

Google Analytics should provide defense against the three attack scenarios introduced in Section 5.2. First step would be to indicate what percentage of visits and conversions had a referrer turned off. Foremost, GA should check the consistency of the information handed over in the HTTP header of the request and in the variables contained in the query string of the `__utm.gif` request and make the result of this check available to the user.

Coping with the third attack scenario, where IP address is forged, would require a redesign of the UTM Sensor. It is suggested that for the first request in the session, UTM Sensor should receive an acknowledgment from Google Analytics Server containing a session verification code. Further requests within the same session should use this verification number. In this way, spoofing sessions containing more than one pageview using a forged IP address would be made much more difficult.

The GAIN software introduced in the following section introduces several security enhancements to Google Analytics tracking. GAIN thoroughly checks the consistency of the HTTP header fields with variables that UTM Sensor sends in the request for the tracking icon and logs the result of this check. It does not aggregate web usage data such as Google Analytics probably does, on the contrary, the data are saved in a relational database whose scheme was designed with respect to the use of the data for data mining purposes.

Chapter 6

Google Analytics Interceptor (GAIN)

Google Analytics is currently a de facto standard web analytics solution. Some sources claim that it is used by some 20% of Fortune 500 companies. This claim would not be difficult to verify, however for illustration purposes a quick survey of ten most visited Czech websites[1] revealed that 20% of them also use Google Analytics (further only GA). Though not statistically representative, this number hints GA' world wide popularity. This chapter presents software dubbed GAIN written by the author, which can be used as a kind of plug-in for Google Analytics. GAIN can be registered with Google Analytics tracking module *Urchin Sensor* to listen for data sent by it. To the best of the author's knowledge this is the first such plug-in provided by a third party.

The goal of GAIN is to provide more granular data than Google Analytics to allow for enhanced fraud detection and data mining. In addition, GAIN puts one of the existing features in Google Analytics to a new use by allowing the website to transmit the semantic data on-line. Google Analytics provide some extensibility what regards the kind of data that it can send to the server. This can be used to sent the semantic information along with clickstream within the existing Google Analytics format and process it with GAIN. Although Google Analytics is not able to read this data, it can be instructed to ignore them.

The structure of this chapter is as follows: the remainder of the chapter introduction explains the reasons for writing GAIN in greater detail. Section 6.1 gives several opening remarks on the process of the actual emergence of GAIN and its overall architecture. Section 6.2 documents GAIN-Listener, the backbone of GAIN. This section expects that the reader is familiar with the Urchin Sensor output format as described in Section 5.1 of the previous chapter. Section 6.3 covers the main ideas of the relational scheme used for data persistence in GAIN. This section also covers space and time complexity issues.

The two Czech Navrcholu Top 10 websites using Google Analytics were e-commerce websites. As it was shown in the previous Chapter, Google Analytics using the present version of UTM Sensor (Google Analytics Urchin Module 1) does not provide enough protection against click fraud and conversion fraud. Moreover, Google is at the same time a major advertising company. There are fears that Google could manipulate data collected by GA in order to boost its profits[2]. Therefore, there is an acute need for a clickstream analysis solution, which

[1] According to Navrcholu.cz web audit - rank list from 26 June 2006

[2] For example, the report of Alexander Tuzhilin [81], who was hired by Google to review its security policy,

could *verify the statistics provided by Google Analytics*. Such a solution should meet the following criteria:

1. Collect data independently of GA
2. Use the same methodology for identifying unique visitors, sessions, new and returning visitors, preventing caching and bypassing proxies.
3. Be able to track visitors on a third party site
4. Retain data on a fine enough granularity to feed data-mining based fraud detection algorithms.

Although there are numerous free and paid software solution which can collect data in a quality way and on a fine granularity, not all are cookie based as GA is. On the contrary, the most widely used free solutions (apart from GA) such as Analog[3] are log-file based and thus they do not meet criteria 2 and 3[4]. Hosted commercial solutions are often cookie-based or they combine cookie and log-file approaches. Their disadvantage is that they do not often provide raw data on a fine enough granularity.

Ultimately, the stumbling block of using *any* existing solution known to the author is that they use a different methodology than GA. The only exception is the Google owned commercial *Urchin Web Analytics*, whose visitor tracking module *Urchin Tracking Module* is a predecessor of *Google Analytics Urchin Module* used in GA. These two modules as of version 6.1 of the former and 1 of the latter do not differ significantly. The tracking module (a javascript file) dubbed UTM Sensor embeds essential visitor information in one string, appends it as a query string to a tracking icon URL and loads the icon.

UTM sensor is patent pending. Other tracking solutions are therefore bound to use at least a slightly different mechanism, which translates into difficulties when comparing the statistics.

6.1 GAIN Architecture

The proposed solution for visitor tracking described in this chapter takes advantage of the ability of the UTM Sensor to send the data for local processing. The visitor data is coded into a proprietary format. Part of it is embedded in a tracking icon query string, the rest is available in cookies. If cookie logging is enabled in the web server configuration, then both the query string and cookies are available in the log file for further processing by the UTM Engine.

The software introduced here allows to capture and process the data sent by UTM Sensor for local processing. In this solution, the tracking icon is not a mere means of recording a

states that "there is a financial incentive for Google not to forgo some of these revenues and simply be easy on filtering out invalid clicks." Further in the report Tuzhilin states that it is unclear to him "why it took Google so long to revise the policy of charging for doubleclicks" [Google applied flawed methodology for counting clicks until March 2005]. Although this report applies to the the Google AdSense advertising scheme, it can easily be observed that some of its conclusions can be valid also for Google Analytics, which is intended to be used in conjunction with AdSense.

[3] `http://www.analog.cx/`

[4] Cross site tracking requires access to the third party log file, which might be difficult to meet

query string and cookies to a log file[5]. On the contrary, a robust server side script is hidden behind the tracking icon. This script dubbed as Google Analytics INterceptor Listener(*GAIN-Listener*) ensures on-line parsing, validation and ultimately persistence of the visitor data. *GAIN-Listener* works in conjunction with *GAIN-DB* – a database vendor specific component and *GAIN-Bridge*. In the current implementation *GAIN-DB* is a layer of several stored procedures. *GAIN-Bridge* acts as a proxy object transferring data from a third-party (usually check out) site to *GAIN-Listener*

This solution does not infringe Google/Urchin patent-pending combination of browser cookies, because it is a mere reader of the data sent by the Urchin Sensor. *Urchin Sensor is not in any way reimplemented or modified*. Part of the Urchin patent deals with efficient data structures for storing the visitor data. However, current implementation of GAIN-DB uses an off-the-shelf relational database. The database scheme was build from the scratch, without any respect to the Urchin patent. Although not as efficient, the proposed relational scheme ensures additional level of validity (due to constraints) of the data while maintaining reasonable space complexity (due to normalization) and availability (data are collected on-line). Finally, the data are ready for analysis with any ODBC compliant analysis or data-mining tool.

Google Analytics INterceptor (GAIN) is a platform-dependent set of modules whose duty is to intercept, preprocess and preserve visitor tracking data transmitted by the Urchin Sensor. In the current implementation, the platforms supported are

- Tracked site: .NET Framework 2.0+ (GAIN-Listener)
- 3rd party site: PHP (GAIN-Bridge)
- Database for persisting visitor data: MS SQL 2005 (GAIN-DB)

6.2 GAIN-Listener (ASPX version)

The largest part of GAIN is the GAIN-Listener with about 2000 lines of code. This component was also the most challenging to write, because it required an understanding of the undocumented format in which UTM Sensor communicates the visitor data. UTM Sensor is approximately 500 lines long JavaScript code written in extremely efficient way, which results into near obfuscated code. With a substantial help of a JavaScript FireBug debugger[6], the code (and its output) was documented.

It should be strongly emphasized that some aspects of the code are quite intricate and although considerable effort was exerted, some of the nuances might not have been well understood. This might have (and probably did) result into bugs in GAIN.

A call to UTM Sensor must be included on each page of the tracked website using the following snippet of code. Setting the `_uacct` parameter (Account) and calling the `urchinTracker()` function is mandatory.

```
<script src="http://www.google-analytics.com/urchin.js"
type="text/javascript"></script>
<script type="text/javascript">
 _ugifpath="http://www.example.cz/trackicon.aspx";
```

[5]Notably, log files are not utilized at all.

[6]`www.getfirebug.com/`

```
 _userv=2;
 _uacct = "UA-135959-5";
 _udn="none";
 _ulink=1;
 urchinTracker();
</script>
```

The _userv=2 tells UTM Sensor to send the visitor data both locally and remotely. The remote target is hard coded, the local one can be set with the _ugifpath parameter. The _udn parameter sets the domain name for cookies, _ulink enables the linker functionality, setting it to 1 allows multi-domain tracking.

GAIN-Listener (trackicon.aspx) is hit by the UTM Sensor each time UTM Sensor hits google-analytics.com. The _ugifpath2 variable is set either at
https://ssl.google-analytics.com/__utm.gif
or
http://www.google-analytics.com/__utm.gif
depending on the protocol used. The hits are initiated from the _uInfo(page):

```
  if ((_userv==0 || _userv==2) && _uSP()) {
   i[ii]=new Image(1,1);
   i[ii].src=_ugifpath+"?"+"utmwv="+_uwv+s;
   i[ii].onload=function() { _uVoid(); }
  }
  if ((_userv==1 || _userv==2) && _uSP()) {
   i2[ii]=new Image(1,1);
   i2[ii].src=_ugifpath2+"?"+"utmwv="+_uwv+s+"&utmac="+_uacct+"&utmcc="+c;
   i2[ii].onload=function() { _uVoid(); }
  }
```

6.2.1 Parsing The Data From UTM Sensor

As it could be seen from the preceding code snippet, the GAIN-Listener hidden behind the _ugifpath URL is not handed over the value of utmcc and utmac parameters. The value of the utmac parameter is not necessary, it can be set as a global parameter for the GAIN-Listener. On the other hand, the utmcc encapsulates the most important variables pertaining to the visitor identification and past behaviour. The values comprising the utmcc can be, however, obtained from cookies.

Completing the data from cookies and the query string is the duty of the LoadAndParseValuesFromQSAndCookies() procedure of the GAIN-Listener. If the same variable is found in the query string and cookies, only the value found in the query string is used (if it is not empty). This procedure also parses the utmcc variable for other variables embedded into it. Curiously, one of the parsed variables (utmz) hides again several others - Extract_Other_UTMS_FROM_UTMZ is used to extract them.

Some variables (utma, utmb, utmc, utmz) are in the "IP" format - they comprise of several numbers delimited by a dot. Unlike IP, each of the numbers has its distinct meaning. There is also one other numerical attribute utmn. SplitUTMintoNumericParts is used to separate the numbers and store them internally as numbers. The consistency of these values is checked by

`CheckAndConvertUTMVariables()` procedure. This check is presently not very extensive, but it is a planned feature to define a regular expression for each of the variables. The variable value would be considered as correct only if it matched the regular expression. This would not,however, solve the problem with several individually correct but contradicting variable values.

6.2.2 Semantic Extension Of Urchin Sensor

The original architecture of UTM Sensor does not allow the webmaster to send any description of the page, such as product price or category. Google Analytics offers only a possibility to additionally group pages based on a regular expression match of page URI. This is insufficient for data mining purposes, modern approaches to WUM rely heavily on provision of semantic information.

Modern e-commerce websites are built dynamically. The building blocks are often the very needed semantic pieces of information that information retrieval approaches try to extract in an additional preprocessing step. GAIN-Listener offers a logical alternative to ex-post provision to semantics to web usage data – the semantics are obtained ex-ante, without any noise, additional tools, and what's most important with minimum effort: the web usage data are semantically enriched during the web-page rendering.

For a dynamic website, server-side web-page rendering is the process during which machine-readable pieces of information are assembled into the human readable form. In this stage, the individual pieces of information are yet isolated and reside in variables. It is usually very easy for the web designer to put these variables into one more use and send it by UTM Sensor the *GAIN-Listener* part of *GAIN*. *GAIN-Listener* makes it possible that all is required is that the web designer constructs a string like the following one

```
OPAString?variablename1=value1&varname2=value2&...
```

and uses it as a parameter for the the call to Urchin Tracker. In fact, she only needs to concatenate a user-friendly name of the variable she wants to attach to the page and its value, both preferably URL encoded. The number of name-value pairs is in no way limited.

The advantage of this kind of providing semantics is that it is absolutely timely. If the price of the product changes, the semantic descriptor changes as well. In addition, it changes only for those visitors that have seen the page with the updated price. The way how these temporarily evolving semantic descriptors are saved is discussed in the following Section (6.3).

How can be Urchin Sensor made to transmit this information, if it does not support it by default? We take advantage of the possibility the UrchinTracker gives us to record actions. This functionality enables conversion tracking, but it is possible to use it also for our purpose. If UrchinTracker is called with a parameter, this parameter is taken as the URL of the calling page. Two or more calls to UrchinTracker may be present on one page.

```
<script type="text/javascript">
 _ugifpath="http://www.example.cz/trackicon.aspx";
 _userv=2;
 _uacct = "UA-135959-5";
 _udn="none";
 _ulink=1;
 urchinTracker();
```

```
 SemanticInfo = "OPAString?price=1200&category=Adventurous%20trips";
 urchinTracker(SemanticInfo);
</script>
```

The first call to urchinTracker logs the URL of the page, the second call sends the semantic information pertaining to it. The GAIN-Listener recognizes that it is not a URI of a page, but a semantic information pertaining to the previous request thanks to the "OPAString" identifier placed in the beginning of the string. Google Analytics custom filter can be used to remove these requests from GA statistics. This filter is based on regular expression URL match, which is in case of OPA variables very easy to specify ("OPAString.*").

6.2.3 Treating E-commerce Tracking

GAIN-Listener is usually invoked by `UrchinTracker` method of the Urchin Sensor. Each request from Urchin Sensor represents either a page view or an action mimicked as a "virtual" page view. Nonetheless, there are some exceptions. If Google Analytics is used for E-Commerce tracking, the following snippet of code might be used to report the details of an e-commerce transaction to Google Analytics.

First, a page should contain a hidden form containing a structured description of the transaction (UTM:T part) and one or more items comprising the transaction (UTM:I).

```
<form style="display:none;" name="utmform">
<textarea id="utmtrans">
UTM:T|[order-id]|[affiliation]|[total]|[tax]|
  [shipping]|[city]|[state]|[country]
UTM:I|[order-id]|[sku/code]|[productname]|[category]|[price]|[quantity]
</textarea>
</form>
```

Next, when it is certain that the visitor completed the transaction, the following code is used to submit the form:

```
<script type="text/javascript">
__utmSetTrans();
</script>
```

The utmtrans procedure sends the content of each item and the transaction summary in a separate request for the tracking icon(s). For example, the following transaction from the official Google documentation is sent in three requests.

```
UTM:T|34535|Main Store|111108.06|8467.06|10.00|San Diego|CA|USA
UTM:I|34535|XF-1024|Urchin T-Shirt|Shirts|11399.00|9
UTM:I|34535|CU-3424|Urchin Drink Holder|Accessories|20.00|2
```

These requests are very similar to requests made for regular (or virtual) page views in the `_uInfo(page)` procedure. Some variables are, however, missing. The most important ones not sent by `__utmSetTrans()` but sent by the standard `urchinTracker()` call are `utmp` (relative page uri) and `utmhn` host name. The transaction type of request uses these semantic variables: `order-id`, `affiliation`, `total`, `tax`, `shipping`, `city`, `state/region`, `country`. The item type uses `order-id`, `sku/code`, `product name`, `category`, `price`, `quantity`.

Transaction line

```
http://www.google-analytics.com/__utm.gif?utmwv=1&utmt=tran
&utmn=461951339&utmtid=%5Border-id%5D&utmtst=%5Baffiliation
%5D&utmtto=%5Btotal%5D&utmttx=%5Btax%5D&utmtsp=%5Bshipping
%5D&utmtci=%5Bcity%5D&utmtrg=%5Bstate%5D&utmtco=%5Bcountry
%5D&utmac=UA-135959-5&utmcc=__utma%3D1.1825175582.118236565
8.1183041360.1183044055.8%3B%2B__utmb%3D1%3B%2B__utmc%3D1%3
B%2B__utmz%3D1.1182365658.1.1.utmccn%3D(organic)%7Cutmcsr%3
Dgoogle%7Cutmctr%3Dpozn%25C3%25A1n%25C3%25AD%7Cutmcmd%3Dorg
anic%3B%2B
```

Item line (may be multiple)

```
http://www.google-analytics.com/__utm.gif?utmwv=1&utmt=
item&utmn=1542426920&utmtid=%5Border-id%5D&utmipc=%5Bsku%
2Fcode%5D&utmipn=%5Bproductname%5D&utmiva=%5Bcategory%5D
&utmipr=%5Bprice%5D&utmiqt=%5Bquantity%5D&utmac=UA-135959
-5&utmcc=__utma%3D1.1825175582.1182365658.1183041360.1183
044055.8%3B%2B__utmb%3D1%3B%2B__utmc%3D1%3B%2B__utmz%3D1.
1182365658.1.1.utmccn%3D(organic)%7Cutmcsr%3Dgoogle%7Cut
mctr%3Dpozn%25C3%25A1n%25C3%25AD%7Cutmcmd%3Dorganic%3B%2B
```

GAIN supports Google Analytics e-commerce tracking. Interestingly, GAIN-DB does not have any special data structures for storing e-commerce transaction data, because it would violate its open-schema principle, which ensures that GAIN can quickly adapt to any format change. Transactions are stored as typed semantic descriptors of a request, which provides the maximum flexibility.

6.2.4 Unescaping Variables

UTM sensor URL encodes the variables that may contain non-ASCII characters. Most variables are converted to Unicode by UTM Sensor prior to URL encoding (escaping). For example, if the tracked page uses windows-1250 encoding, its title "Mont Keňa" is stored in the `utmdt` variable as "Mont%20Ke%C5%88a", which clearly corresponds to a conversion of the "ň" character to its unicode counterpart.

Unfortunately, the UTM Sensor cannot convert to Unicode the variables, which already arrive URL Encoded and are in some other encoding, such as windows-1250. This is the case with the `utmctr` variable, which contains the term used by the visitor to find the tracked website on a third party search engine. Search engine result page usually contains the sought keyword in its URL. If the search engine uses 8 bit character set, which for example `atlas.cz` does (as of 27 June 2007) and the page URL contains not ASCII characters, there are URL encoded in only 2 bytes (plus leading percent sign). Because the referring URL no longer contains non-ASCII characters, UTM Sensor essentially only checks if the referring site belongs to its search engine list. If it does, it tries to extract the query part of the referring URI and replacing white spaces with a plus sign (+) it puts it as `utmctr` (part of utmcc) to the query string.

Because there is little means of knowing what character set the each referring search engine uses, a heuristic function `UniversalUnescape` designed by the thesis author is applied in the GAIN-Listener in order to correctly decode the parameters. A straightforward use of `System.Uri.UnescapeDataString` method available in .NET 2.0 is not sufficient, because it works only with Unicode escaped strings.

As it was explained, we expect the original character to be either one of the ANSI variant or Unicode. ANSI characters are URL encoded in a very similar way to Unicode characters. The difference is in the number of hexadecimal digits following the percent sign. There are always two when ANSI character set is used. For Unicode/utf8 non-ASCII characters are represented by four and more hexadecimal digits. The `UniversalUnescape` first tries unescape to Unicode. If the conversion fails, an attempt to unescape to ANSI comes to place. The odds of incorrectly unescaping ANSI coded string as Unicode are very low. The specific ANSI variant used during the second unescape attempt is currently hard coded in the `trackicon.aspx` and is set to Code Page 1250 (ANSI - Central Europe).

6.2.5 Error Handling

Detection and handling of errors and warnings is a key function of GAIN. The purpose of GAIN is a verification of the statistics provided by Google Analytics. GAIN should be able to record suspicious activity. It is designed not to merely discard requests with missing mandatory information or on the other hand blindly accept a request with contradicting or missing variable values. If an error occurs in the GAIN because of an unexpected combination of input variables, this incident is meticulously recorded, because it might either indicate a hit inflation attack, or conversely a hit unrighteously omitted from GA statistics.

There are three levels of error handling/checking in GAIN:

1. Presence of mandatory and optional variables and their formal consistency.
2. Consistency of selected variables with values obtained from the browser.
3. Uniqueness of the used user ids, temporal consistency of data.

The first two levels are checked in GAIN-Listener, the last one requires availability of past requests, therefore it is left on GAIN-DB. Every identified problem is assigned a severity level, a brief explanation and a trace containing the values of relevant internal variables and finally it is recorded to the database and/or log-file.

A lot of effort was invested into designing both machine and human readable error log file format. An example error log is present on the Thesis CD.

In addition to referential integrity constraints, GAIN-DB undertakes additional checks. E.g. GAIN-DB checks, if a visitor request ID already exists in the database. If it does, it is considered suspicious and recorded. During each execution of UrchinTracker, a new pseudo Random Request ID is generated. Request ID is populated uniformly from a relatively broad interval, which should for websites with small and medium traffic ensure that the same ID is not generated within a reasonable time span.

In the database, the errors are kept tied to the particular visitor request during which the error occurred. Fatal errors, which prevented the visitor request to be recorded into the database, are assigned a random negative number as a placeholder for nonexistent request id. Error types are unified in order to be usable in a further quantitative data-analysis.

The log-file is primarily intended for logging serious errors. It provides detailed information about the error and the visit which caused it. The GAIN settings offer several options for tuning the behaviour of the error handling facility.

6.2.6 Tracking Icon

GAIN-Listener has two possible return types. It either generates a 1×1GIF file with minimum possible size of 35 bytes. It is not possible to achieve such a small image size with the image library of .NET 2. For performance reasons, it is also not viable to keep this file on disk and load it every time the track icon is requested. Instead, this image is hard-coded as an array of bytes directly into `trackicon.aspx` and output as a binary stream to the HTTP client. The rendering is thus almost as fast as it can possibly be.

The second output format is for debugging purposes. It is a JPEG image with selected debugging info drawn upon it.

6.3 GAIN-DB (MS-SQL 2005)

GAIN-Listener sends visitor data (and possible errors) to GAIN-DB, which is in the current implementation a layer of two T-SQL stored procedures – `AddRequest` and `AddError`. These procedures are relatively simple. The input `AddRequest` receives from GAIN-Listener is already well preprocessed, its task is to save the visitor data to a relational database scheme in a third normal form.

The return value of the procedure is the ID code GAIN-DB assigned to the request. This code can then be optionally used by GAIN-Listener for calling the `AddError` procedure.

The scheme comprises of approximately 20 tables. There are several core tables, the rest are code tables. The core tables are Visitors, Visits, Requests, Errors and Fatal_Errors.

Each successful execution of AddRequest adds a record to the Request table. Request table is the table with the largest number of rows in the schema and it quickly becomes the largest table what concerns the amount of space occupied. It contains the following columns: ID, Request, VisitID, URIID, TimeAdded and IPID. The column names are self-explanatory, except for the Request column, which contains the random request id generated by UTM Sensor.

6.3.1 URI Handling

Code pages are important not only because they save space, but also because they help to retain consistent data entries, which is important for data-mining applications. The highest attention was paid to correctly handling URIs. URIs are decomposed into the host, relative path and query string part, each saved in a separate table.

One problem is posed by the feature that allows the webmaster to record a virtual page view, which is intended for recording user actions other than pageviews. The downside of this is that the data sent by Urchin Sensor does not allow to differentiate what is a pageview and what is an action. As a consequence, reported user actions are handled like if it were URIs of pageviews. There is one exception – the semantic information sent via the OPAString see Section 6.2.2 is identified and handled differently.

No distinctions in storage is made between URIs coming from the tracked site (further also internal URIs) and referring URIs (further also external URIs).

6.3.2 Semantic Information

The database schema supports appending semantic information only to requests, not directly to pages in spite of the fact that semantic information pertains mostly to a page. There were two reasons for this decision:

1. It is desirable to process UTM Trans requests in the same way as regular requests
2. The semantic information attached to one URI may change in time

UTM Transaction requests (information about e-commerce transaction) do not contain URI but can principally carry additional semantic information, therefore it was necessary to assign the semantic information directly to Request.

For regular (`urchinTracker()`) requests, assigning semantic information to request allows for capturing the fact that one URL can have different semantic content (such as product price) at different times. There is unfortunately some sacrifice in the efficiency of storage, because the identifier of the semantic information is stored in the `Request_SemanticIdentifier` Table for each request for a page instead of being stored once for each URI in the URIs table. Each Request can be assigned multiple semantic identifiers of a different type. Types are stored in `SemanticIdentifierClasses` Table, the actual values are stored in `SemanticIdentifiers` table.

Due to use of open schema, the addition of a new type of semantic identifier is fully automatic and does not require any effort from the user or a modification of database schema. The `AddRequest` procedure was designed with extensibility in mind. The mandatory information is handed to it as required parameters, but the semantic information is handed over as a XML variable. This allows request to be described by any number of semantic variables. In addition, the semantic type of the variable can be specified as well. Due to open schema, all variables are internally stored in one table as `nvarchar` data type

The additional semantic identifier can be added either a priori by UTM Sensor or a posteriori in optional data enrichment step. Examples of semantic identifiers assigned automatically by the UTM Sensor to an external and internal URI could be `Sought term` or `Page Title` respectively. The OPAString (see Subsection 6.2.2) can be optionally used to assign additional semantic information.

A domain specific taxonomy or text mining software might be used to invent new internal semantic identifiers for internal URIs. External URIs might be (indirectly as explained above) assigned for example their page rank value.

6.3.3 User Agent Related Information

The code tables for the various visit attributes form another group of database tables: Flash client versions, IP addresses, preferred user (browser) languages, screen resolutions, and user agents. The last attribute is a bit more complex than the previous ones, because the user agent can be typically decomposed into several distinct variables – Netscape compatibility version, browser name, browser version, OS, OS version, Extensions.

Unfortunately, there is no unified format for user agent stings, each browser vendor uses a slightly different formatting. Due to fast development of browsers, it would be cumbersome to perform user agent string analysis on the GAIN-Listener level due to necessary frequent

updates of the browser definitions. It is neither viable to perform the analysis in the GAIN-DB, because of the poor performance of T-SQL string functions. For the time being, user agent string analysis and decomposition are left for further processing.

6.3.4 Error Persistence

Finally, GAIN DB exposes one unified interface to GAIN-Listener for reporting of both critical errors and warnings. A critical error is usually raised, when some mandatory information handed over to GAIN-Listener by the UTM-Sensor is missing or of a wrong data type. When critical error occurs, `AddRequest` call is skipped and GAIN-DB calls directly the `AddError` procedure. For less serious errors, `AddError` is called after `AddRequest` with the `AddRequest` return value –the internal database request id – as one of the input parameters. The `AddRequest` procedure stores the error details either in the Errors or Fatal errors table depending on the value of the request ID parameter. If it is a new error type, it is added into error types code table.

6.3.5 Space Complexity

Visitor data are saved to a relational database in a third normal form. Storing integer value in MS SQL takes 4 bytes, small integer 2 bytes and tiny integer 1 byte. Datetime value requires 8 bytes. Most tables use a datetime column to record when a row was added to the database. All tables have an identity integer column. Boolean values are stored as bit datatype and require only 1 bit. Nchar requires two times the number of bytes entered + 2 bytes. Bit columns are stored together as 1 byte if there are 8 or less bit columns in a table.

The number of bytes needed to store information pertaining to the *visitor* is ID (4 bytes) + GAVisitorID (4 bytes) + Time (8 bytes) + NoCookieVisitor + NoReferrerVisitor (1 byte). Altogether 17 bytes.

The *Visit* table contains five integer columns, four small integer columns, one tinyint column and a datetime column. Altogether $5 * 4 + 4 * 2 + 1 * 1 + 8 = 38$ bytes.

The *Request* table contains five integer columns, one nvarchar(1) column and a datetime column. Altogether $5 * 4 + 1 * 2 + 2 + 8 = 32$ bytes.

The minimum amount of space needed to save information pertaining to one pageview of a new visitor is $17+38+32 = 87$ bytes. Here we assume the all the information referenced (such as the IP address or user agent string) is already present in the code lists and no semantic information is attached to the requests.

Attaching one piece of semantic information to the request costs $3 * 4 + 8 = 20 bytes$ provided that the semantic information is already in the database.

Storing information about the average case - first time visitor who uses a standard browser, is hidden behind a large proxy[7] and views three pages, each described by one common semantic variable value thus consumes $17 + 38 + 3 * 32 + 3 * 20 = 211$ bytes. A 100 MB database can[8] store about one million visitor requests (including the overhead for the code lists). For a website with 2000 visitors a day and average of 3 requests and 1 visit per visitor, the capacity of the database should suffice for approximately 400.000 visitors or 200 days of traffic.

[7] It is assumed that the proxy IP address is already in the IP code table.

[8] With the code tables occupying one fifth of the database

6.3.6 Time Complexity

Precise evaluation of the speed complexity is beyond the scope of this work. The empirical testing showed that the execution time of GAIN-Listener and GAIN-DB was in between 200 - 750 ms, while the Google based data collector executed in less than 200 ms. The better performance of Google data collection can be attributed mainly to the following (in the order of the presumed influence):

1. Google uses hash not relational tables for storage
2. Google apparently performs little consistency checking
3. Google perhaps does not retain all the detailed info as GAIN-DB does

Chapter 7

Association Rule Mining And Clustering On A Real World Dataset

E-commerce applications of clickstream analysis put emphasis on conversions and referrers. Conversion basically distinguishes visitors who turned into money from those who did not, referrers indicate which promotional activity got these visitors on the site.

This chapter presents an empirical evaluation of several datamining algorithms on a real world e-commerce dataset. The goal of the mining was to find interesting pieces of knowledge which could be used by marketing professionals to better understand the browsing patterns of the website's visitors, identify visitor segments, and increase the Return On Investment of advertisements.

Two existing association rule mining systems *Arules* and *LISp-Miner* were employed. The visitor segmentation was done using three variants of the K-means clustering algorithm, which were coded specifically for this purpose by the author of this thesis. For details of these systems and algorithms please refer to Chapter 4, where their most important features are explained. Additional information is available in the official documentation of the *Arules*[1] and *LISp-Miner* systems[2].

The primary goal of this chapter is to demonstrate some of the preprocessing and data mining algorithms introduced in the theoretical part of this thesis on real world datasets. Two major data mining techniques are used – association rule mining and clustering. Multiple implementations of each of this classes of algorithms were executed on the same dataset, which allowed for cross-algorithm evaluation.

The structure of this chapter is as follows. The data and the preprocessing phase are described in Sections 7.1 and 7.2. The preprocessing comprises of visitor profile construction phase and discretization phase. The former is common to all algorithms used in this chapter, the latter was carried out only for associations and differently for each of the implementations. The description of the experiments with the association rule mining algorithms follows in Section 7.3. The part of this chapter on association rule mining is concluded with the performance comparison in Subsection 7.3.7, encountered problems are discussed in Subsection 7.3.8. Experiments with clustering algorithms are described in Section 7.4 followed by

[1] `http://cran.r-project.org/src/contrib/Descriptions/arules.html`
[2] `http://lispminer.vse.cz`

an evaluation of the algorithms in Subsection 7.4.1. The concluding Section 7.5 brings the evaluation of results from the business point of view and some remarks on when to apply which algorithm.

Acknowledgment The experiments with the *Arules* were carried out during the author's study stay at the Faculty of Informatics and Statistics of the Slovak Technical University in Bratislava (STUBA) by Matej Minarik (STUBA) and the Tomáš Kliegr (FIS VSE).

Terminology note In this chapter, we will use the terms visit, session and visitor interchangeably. The reason is that in our test data, every visitor had exactly one visit. Almost the same approach could be however used for visitors with multiple-visits; only the Score formula would have to be changed in order to reflect the visit order number. Therefore, we opted to use mostly the more general term visitor and its collocations (such as visitor profile) throughout this chapter.

7.1 Data Description

The data used during mining were collected with the predecessor of GAIN (see Chapter 6) from a website of a Czech tour operator. The exact methodology of data collection is described in the author's bachelor thesis [53]. The approach to collecting clickstreams was "cookie-based" as opposed to more traditional log-file based approaches. The application used for visitor tracking identified the requests coming from individual visitors using cookies and stored their click-streams already in a sessionized way into a relational database. There was thus no need for applying heuristics to identify sessions. As the tracking was triggered by the visitor's browser downloading an image, this technique also automatically removed (ignored) vast majority of traffic coming from crawling bots.

The data were then merged with additional semantic information describing the structure of the website and the content of the individual pages using a service and content taxonomies. The result of this operation are the following three attributes attached to each page view: `Topic` (Content) and `Cat` and `ECAT` (Service on two levels of granularity). In addition, each referring website was assigned a lower granularity class, this assignment is stored in `search_engine_map.csv`. A heuristic based on time on page and page order was used to assign weights to each click (`PageScore` attribute). The Score heuristic (see Section 3.7.1) was used to calculate the weights. For convenience sake, a quick recapitulation of the heuristic is included here.

$$S = (\ln(o) + 1) * t \tag{7.1}$$

The logic behind this heuristic is that the relevance of the page to the visitor's information need rises with the time on page t and with the order of the page in the visitor's clickstream o. The logarithm in the equation ensures that the variance of the values of Score remains in reasonable boundaries and that it provides some robustness against outliers in o. Outliers in t were detected and removed beforehand.

The aforementioned preprocessing steps were conducted using SQL as described in [53] and then exported into two files. The file `visitors.csv` contains one row per session with columns denoting attributes common to all page views in the session. The file `clicks.csv`

Visitor ID	PageName	Cat	ECAT	Topic	t	S	o
1	www.poznani.cz	Info	Homepage	Obecne	30	30	1
1	zajezdy-se-stany	Hledani	Katalog	Turistika	30	50	2
1	horska-turistika	Hledani	Katalog	Horska turistika	30	62	3
1	expedice-hory	Hledani	Katalog	VHT	30	71	4
1	voda	Hledani	Katalog	Voda	30	78	5
1	rafting	Hledani	Katalog	Rafting	30	83	6
2	norsko-....htm	Zajezd	Neni	Turistika	300	300	1
2	narodni....htm	Zajezd	Neni	Turistika	30	50	2

Table 7.1: Data for two visitors from clicks.csv (adapted)

contains information specific to individual pageviews including the page URL (`PageName`) and the already explained `Topic`, `Cat`, `Ecat` and `Score` attributes (columns).

The input data consisted of 15560 visits and 38543 pageviews. The aforementioned files `clicks.csv`, `visitors.csv`, `search_engine_map.csv` are available on the thesis CD.

Plots of some of the important attributes present in the data are available in the Appendix.

7.2 Data Preprocessing

The input for the association rule mining and clustering algorithms used here is one data matrix, in which columns represent attributes and rows visits (sessions). That is each visit is represented in one vector (a row in the physical representation[3]) using a given set of attributes. We call this row for a visitor i a visitor profile UP_i. Some theoretical introduction to the notion of visitor profile was given in Chapter 3.

Visitor profile should contain attributes common to the entire session (e.g. `Referrer`, `Day`, `Hour`). We will refer to these as *common session attributes*. In addition, the visitor profile should contain the individual visited pages represented by URL `PageURL` and their associated `Topic`, `Cat` and `Ecat` attributes.

However, we need to solve the problem discussed in Chapter 3 prior we can define the *visitor profile* more precisely. All our algorithms require fixed-length input data, but our data are variable-length (sessions comprise of variable number of pageviews). This problem applies only to the information pertaining to pages, not to the *common session attributes*.

7.2.1 Page Vectors

The solution taken is to represent pages as *multimodal* page vectors proposed by [25]. We have modified the components of the page vector and use different features than in [25]. The result is a custom page vector P_d which represents the page d. In our approach, P_d has the same number of attributes for all visits d and it comprises of a less granular expression of page content (`Topic` sub-vector), and the expression of the type of page across the Service dimension on two granularity levels represented by `Cat` and `Ecat` subvectors.

Please note that from now on `Topic`, `Cat` and `Ecat` are *sub-vectors*, not attribute value pairs. Each of the attributes used to describe a page is represented by a subvector

[3]In the following text, we will sometimes use the term row when its clear from the context that it means the visit vector.

Visitor ID	Hledani	Info	Nezjisten	Prihlaska	Zajezd
1	344	30	0	0	0
2	0	0	0	0	350

Table 7.2: CAT subvector of the visitor profile (Example for two visitors)

of the page vector. The number of elements of the particular subvector corresponds to the number of distinct values of the attribute. In the terminology of [25], these subvectors are called *feature spaces* or *modalities*.

Page URL subvector `PageName` is not included, because it contains unnecessarily granular information for clustering. For association rule mining, using the page URL's in our experiment settings lead to extremely high running times, therefore we decided not to use this information. However, the original approach [25] contains the URL subvector.

For example, lets for the moment consider that we nevertheless decided to include the `PageName` subvector to the page vector. The subvector `PageName` contains M elements, where M is the number of distinct pages (URLs) on the website, each component of the vector corresponding to one distinct URL. For page d, the value of all but one element of this vector is 0. Value 1 is placed in the element corresponding to the URL of the page d. Hence, `PageName` is unit-length. In our test data, all page sub vectors were unit-length. However, it may not be – in principle – always so.

7.2.2 Visitor Profile

Having explained how page vectors were created, we can finally proceed to the definition of *visitor profiles*. Visitor profile UP_i is a multi-modal vector that contains the same subvectors as the page vector, and in addition to it a subvector representing the common session attributes.

The first part of the visitor profile UP_i is constructed as a linear combination of page vectors P_d of pages $d = 1, 2, \ldots, N$ appearing in the clickstream of visitors i. The coefficients of the linear combination are the weights s_id. In our implementation, the weight s_{id} is a Score of the particular page in the clickstream of the visitor i.

$$UP_i = \sum_{d=1}^{N} s_{id} P_d \tag{7.2}$$

The second part of visitor profile, which represents the common session attributes, can be viewed as one subvector, whose components correspond to `Day`, `Hour`, `SearchEngine`, `katalog`, `prihlaska` and `sleva` attributes.

The last three attributes make up in a sense for the omission of the `PageName` subvector. In fact, they constitute the important points on the path as proposed in Section 3.6. These boolean attributes contain a true/false value indicating, whether the visitor requested a hard copy of a trip brochure, submitted an application or was interested in the discounts provided. If any of these actions is true, the visitor contributed to increasing the ROI of the advertising campaign, which attracted her on the website.

The SearchEngine attribute contains a less granular expression of the visit referrer, based on a code table `search_engine_map.csv`. This attribute was used instead of the more granular `Referrer` attribute.

Visitor ID	SearchEngine	Day	Hour	katalog	prihlaska	sleva
1	dir.seznam.cz	Sunday	23	0	0	0
2	search.atlas.cz	Sunday	23	0	0	0

Table 7.3: Common visit attributes part of visitor profile (Example for two visitors)

At the first glance, it may seem that the information contained in the order of the visited pages is lost. Visitor profile expresses only whether a certain page has been visited or not. However, this information is partly retained due to the fact that Score is used to weight the attributes - and the order of the page is incorporated into Score.

Preprocessing Implementation

The construction of the page vectors UP_i was conducted in the R environment, the source code is contained on the thesis CD as `preprocess.R`. This procedure has to be run once for every page subvector, which needs to be processed. The second part of the file `preprocess.R` contains the routine, which outputs the common visit attributes.

In line with the common practice, we excluded short visits (of length 1 and 2) from further processing in the beginning of the algorithm. After the short visits were removed, *6644* visits remained.

The matrix, which represents the visitor profiles, contains a column for each possible value of attributes CATegory, Extended CATegory and Topic and one column per each of the common visit attributes. However, physically the script outputs this matrix into several files.

7.2.3 Discretization

The next step of the preprocessing was the discretization of attributes. This step was not needed for clustering, because clustering takes the numerical character of data to its advantage.

Clustering was performed differently in *Arules* and in *LISp-Miner*. In *Arules*, we wrote a function `discretize.R`, which discretizes attributes containing PageScore into four values `none`, `small`, `medium` and `high`, corresponding to intervals $< -1, 29)$,$< 29, 80)$,$< 80, 200)$ and $< 200, 2000 >$.

In *LISp-Miner* we used program LMDataSource to bulk process these attributes. The input data are stored in `trans_data.mdb` database. It was not possible to use predefined intervals for attribute discretization in the bulk mode (although it is possible in the manual mode), so we chose "equi-frequency intervals" and a number of maximum intervals generated. For most attributes 2, sometimes 4 categories were created. The output of the transformation was stored in a proprietary format (LISp-Miner metabase) in `LMzz.mdb`.

The remaining common session attributes `SearchEngine`, `day`, `hour`, `katalog`, `prihlaska` and `sleva` were transformed to class factor and added to the matrix. The attribute hour was discretized into values `morning`, `afternoon`, `evening` and `night`. The values of `SearchEngine` were categorized into 5 values according to a predefined mapping. The resulting matrix `vys_data.csv` was ready for use by the apriori algorithm from the *Arules* package.

Naming convention To better distinguish which item is derived from which original attribute, we used the following item prefixes in *Arules*: `ECN_` for `ECAT`, `CN_` for `CAT`, `TN_` for

`TopicName`. We used slightly different prefixes in LM and one additional prefix K_ for conversion.

7.3 Associations – Experiments

The goal of the experiments was to assess the two data-mining systems used – *4ft-procedure* of the *LISp-Miner* system and *Arules package* of the *R Project for Statistical Computing*. *R* is a free software environment for statistical computing and graphics[4]. The comparison of the two systems is of interest, as their underlying origin, theory, algorithms and user interface differ substantially. Our intention was to test, whether these two systems are fundamentally capable of solving the same elementary tasks with the identical outcome and in a comparable time while maintaining reasonable memory usage.

The experiments were run with *LISp-Miner* v. 8.07.00 of February 2007 and R 2.4.1. of 18 December 2006 with *Arules* 0.6-0 of 12 March 2006.

Please refer to Chapter 4 for an explanation of the meaning of the various parameters, statistical measures and settings used in this section.

7.3.1 Arules Experiment No. 1

The following settings of the Apriori algorithm were used in the first experiment: minimum support was set to 1%, minimum confidence to 0.5%, an additional rule evaluation measure – Absolute difference of improvement to 1 (AIMP) was used with a minimal value set to 0.5, the minimum rule length was set to one and the maximum to 5. Only rules with one of the items `prihlaska=1`, `sleva=1` and `catalog=1` on the right-hand side were mined.

For time complexity reasons, only the first thousand of visits was used. The algorithm with these settings generated 75 231 rules. The rules were ordered by their confidence and only first several thousands of them were listed for inspection. Most of the rules had confidence equal to 1. The reason was that there was at least one of the items $CN\_prihlaska = low|medium|high$, $ECN\_prihlaska = low|medium|high$, $ECN\_katalog = low|medium|high$, $ECN\_slevy = low|medium|high$ on the left side of the rules. These corresponded to items $prihlaska = 1$, $sleva = 1$ or $catalog = 1$ on the right-hand side (RHS) of the rule.

7.3.2 Arules Experiment No. 2

The items $CN\_prihlaska = low \mid medium \mid high$, $ECN\_prihlaska = low \mid medium \mid high$, $ECN\_katalog = low|medium|high$, $ECN\_slevy = low|medium|high$ were removed in this experiment to prevent the generation of uninteresting rules, which occurred in the first experiment. The same parameter settings as in experiment no. 1 were used except for the following: the minimal value of the absolute difference of improvement to 1 (AIMP) was set to 3, and maximum length of rules was set to 3. All the visits were used in mining.

Three rules were generated with these settings. The item $ECN\_Homepage = small$ was on the left side of each of them together with one of $CN\_Info = high|medium$ or $CN\_Hledani = none$. The support of these rules reached values from 1.1-2.9 %, confidence was between 0.51 and 0.77 and AIMP between 5.1 and 7.8.

[4]For more information see the homepage of the project at `http://www.r-project.org/`

Rule side	Allowed items
LHS	all attributes from the ECAT, CAT, Topic, Referrer and Time groups
RHS	Conversion attributes K_katalog, K_prihlaska, K_sleva

Table 7.4: Task settings: Rule appearance

The experiment was repeated with maximum rule length set to 4. 158 rules were produced in this case. The rules closely resembled the preceding three rules, because the left side of each of them contained two items from the previous rules with different third additional item.

These additional items were not useful, which could be observed from the fact that the interestingness measures have not generally increased with confidence of only a few rules growing up by an insignificant amount. This was not surprising, because the added items belonged among the group of items, whose confidence in the whole dataset was close to one. An example of such an item is $CN\_Alpy = none$.

One important fact about the rules found in this experiment is that the right side of the rules was the same for all the rules – it contained the item $katalog = 1$.

7.3.3 Arules Experiment No. 3

So far no rule was found which would have the very interesting items $sleva = 1$ or $prihlaska = 1$ on the RHS. Therefore, we decided to seek only for such rules in this experiment. To achieve that, we decreased the minimum support to 0.1%.

We started with maximum rule length set to three. Subsequently three rules were generated, item $ECN\_Pojisteni = medium$ was on the left side of each of the rules together with one of $CN\_Info = High$, $TN\_Obecne = high$ or $CN\_Info = medium$.

Next, the maximum rule length was increased to four. In that case 89 rules were discovered. There was not much new information. Similarly as in the previous experiment, the right sides of most of rules with high values of all interest measures contained two items from the shorter rules with an additional uninteresting item.

Again only one item – $sleva = 1$ was on the right side of the rules. We had to decrease the minimum value of support to 0.01% to obtain rules with $prihlaska = 1$ on the left side. Although these rules had confidence equal to one, they were useless, since their support was too low - it corresponded to only a single visit from the whole dataset.

7.3.4 LISp-Miner Experiment No. 1

We used the following setting for rule appearance:

LHS For ECAT we specified length interval $< 0; 2 >$, for CAT $< 0; 1 >$, for Topic $< 0; 2 >$, for Referrer $< 0; 1 >$ and for Time $< 0; 2 >$.

RHS RHS is constituted by only one cedent of the length $< 1; 2 >$. First we used support threshold 30 and confidence 20%. No rules were found for this setting for tens of minutes, so we interrupted generation and used weaker settings - lowering support threshold to 10 and confidence to 10%. After 9999 rules were found the generation of rules stopped due to *LISp-Miner* constraint.

After having observed problems in the first attempts, we used a different quantifier – Double Founded Implication with minimum value 0.4 together with the minimal absolute support of 30. 238 rules were generated in 10 m 38s. We sorted the rules according to the recommended *DConf* criterion. All generated rules were obvious ones with $Dconf = 1$.

The last configuration we tried was with absolute support threshold 30 and Above Average Implication (AAI) threshold 1.5. After 7 hours 414 hypotheses were found. When sorted according to the value of AAI, the most interesting hypothesis (ranked 11.) was:

$$SearchEngine(dir.seznam.cz, www.holan.cz, www.trekking.cz) \& \; Hour(rano) \Rightarrow K\_Katalog(1) \quad (7.3)$$

7.3.5 LISp-Miner Experiment No. 2

Experiment 2 was identical to *Arules* experiment no. 2. Unlike in all other experiments performed with LM, we used the `vys_data.csv` as an input matrix. The input data were loaded to a database (`dataR.mdb`), discretized in the same way as in R and stored in `LMZZr.mdb`. The coefficient type was "subset 1-1" in all cases. *4ft-Miner* generated fewer rules than *Arules*, because it automatically excluded from LHS the items, which also (more reasonably) appeared on the RHS.

7.3.6 LISp-Miner Experiment No. 3

Last experiment was carried out in *SD4ft-Miner* procedure of the *LISp-Miner* system as briefly explained in Section 4.3 and in more detail in [66]. We tried to find such disjunct sets of visitors α, β both defined by `Hour` and `Day` attributes which differ significantly in the relation of `ECat` (being on LHS) and `Konverze` (RHS) under the condition containing attributes `TypOdkazovace` and `SearchEngine`.

The length of LHS was $< 0; 3 >$, RHS $< 1; 3 >$, first set and second set $< 1; 2 >$, condition $< 1; 2 >$. We specified the minimum a frequency of First set and Second set to be 5, and the difference of value of confidence of both sets to 0.2.

828 hypotheses were found until we interrupted generation after 11 hours. Perhaps the subjectively most interesting rule was

$$lastminute < 30; 3051 > \& \; TopicNeni < 0; 30 > \& \; TopicObecne < 120; 6803 > \Rightarrow$$
$$K\_Sleva(1) : Hour(Rano) \times Hour(Odpoledne, Vecer)/TypOdkazovace(Fulltext) \quad (7.4)$$

with *Dconf* 0.23 and frequencies 8 (first set), 5 (second set).

According to this rule, visitors interested in lastminute holidays coming from fulltext search engines will check more likely additional information about discounts when they come in the morning than in the afternoon or in the evening.

7.3.7 Arules and *LISp-Miner* Performance Comparison

We decided to repeat the experiment no. 2 executed in *Arules* with *4ft-Miner*. We did so to confront somehow the performance of both systems. We run the same mining task with the same settings in *4ft-Miner* and *Arules* on the same machine. The experiments were run on IBM R50e with Intel Celeron-M 1.4 GHz processor and 756 MB of PC2700 DDR 266 MHz RAM and 7200 rppm IDE HDD on Windows XP SP2.

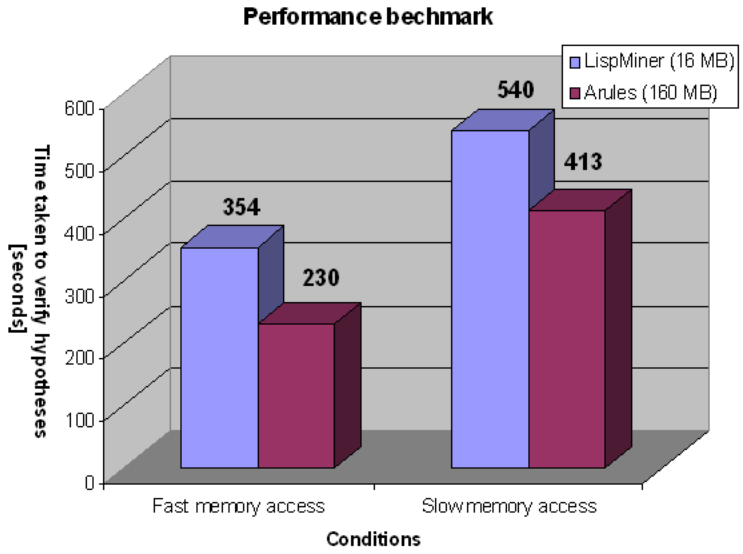


Figure 7.1: Experiment 2 with max rule length 5. LISp-Miner seems to scale better than *Arules*. Lower value is better

The computation in *4ft-Miner* took a little bit more time, 27 seconds compared to 23 seconds it took to *Arules*, in the case of rules of length four. We received slightly different results. *4ft-Miner* generated only 142 rules compared to 158 rules produced by *Arules*. The difference was caused by the fact that *4ft-Miner*(unlike *Arules*) does not generate rules that contain the same item both on the right side of the rule on its left side. *Arules* generated 16 rules such rules – with one of items $prihlaska = 0$ or $sleva = 0$ on their right-hand side. Except for this difference the rules generated by R and *4ft-Miner* were the same. Identical were also the computed values of the few interest measures which were present in both the systems.

During our use of the R system we noticed that the environment tends to allocate a lot of memory. For larger problems it even seemed to freeze. We did not notice such behaviour when using *4ft-Miner*.

To get a better understanding of how the two systems scale, we ran the experiment 2 for rule size 5. We carried out this experiment under two computer configuration – in the first case, the systems had enough free physical memory, in the second case, there was only a minimum of free RAM available and thus disk swapping occurred.

As we can see from the graph depicted on Figure 7.1, *Arules* outperformed the *4ft-Miner* procedure of the *LISp-Miner* system speedwise in both settings. However, the memory usage of R increased dramatically, when *Arules* was run. The speed of memory access seems to be substantially more critical for R than for *LISp-Miner*. We assume that for real world datasets, *LISp-Miner*'s conservative approach to memory consumption might ensure it a performance advantage over *R*'s *Arules*.

It should be noted that we tested *Arules* with the option "maximize speed setting". We tried also the 'minimize memory usage setting", but *Arules* consumed the same amount of memory as with the former setting and the task did not finish, because R crashed.

7.3.8 Encountered Problems

Both reviewed data mining systems were able to cope with the amount of data processed. We did not run into any unsurmountable obstacle when using either *LISp-Miner* or R.

Perhaps the biggest systematic problem which occurred were redundant items in the generated rules. By redundant we mean that some items (attribute-value pairs in *Arules*, literals in *4ft-Miner*) appeared virtually in all the rules due to the fact that they were present in almost all visits. A typical example is *Alpy*(*none*), which was present in all visits, except for a handful of those with a page with trips to the Alps in their clickstream.

Remedy to this problem in *Arules* was to remove these "nothing telling" attributes (here *Alpy*(*none*)) in the Appearance setting. In *LISp-Miner*, it is possible to use a similar solution as well by removing the unwanted attribute value in the preprocessing *LMDataSource* procedure. However, we used a different solution – we used coefficient of type "right cut" of length $N - 1$, where N was the number of categories of the attribute. In this way, it was assured that the leftmost category (here None) will not appear in any rule.

Another problem posed too obvious rules. We did not find a way how to limit the *maximum* value of confidence or of any other quantifier[5] in either of the systems. The too obvious rules could be removed only in an additional rule pruning step.

Overall, we conclude that *LISp-Miner* provides better functionality and stability than *Arules* while maintaining comparable performance. What we found rather limiting on *LISp-Miner* was the fact that it has a quite strict threshold for the maximum number of generated rules - 9999. A minor disadvantage is that although one can set a minimum and maximum length of a partial cedent, a minimum and maximum length of the entire LHS or RHS can not be specified.

7.4 Clustering – Experiments

The compared algorithms were already introduced in the Chapter 4. It is the *spherical K-means*, which is sometimes also referred to as "vector-space variant of K-means", *repeated bisection K-means*, and a proposed modification of the latter – *repeated bisection K-means with initialization using generic algorithm*. The program implementing all the three clustering algorithms (see also Figure 7.2) as well as the test data and configuration file are included on the thesis CD.

The goal of the experiments was to compare the suitability of the three clustering algorithms for web usage mining, namely for visitor segmentation. All the experiments were carried out on the subset of data presented in Sections 7.1 and 7.2.

The full number of visitor profiles after the short visits were removed was used (6644). However, only two feature spaces were used – the feature space of the Topic attribute (dimension 27) and of the Cat attribute (dimension 5). Two modalities with weights 0,6 and 0,4 respectively were used to represent these attributes.

The modalities are specified in the configuration file `modalities.csv` by means of specifying the number of the first and last column belonging to the modality and the weight assigned to the modality, one modality per row of the configuration file. The weights must sum up to 1 and one column must not be included in multiple modalities.

[5] Except for support. Maximum support can be specified in *LISp-Miner*.

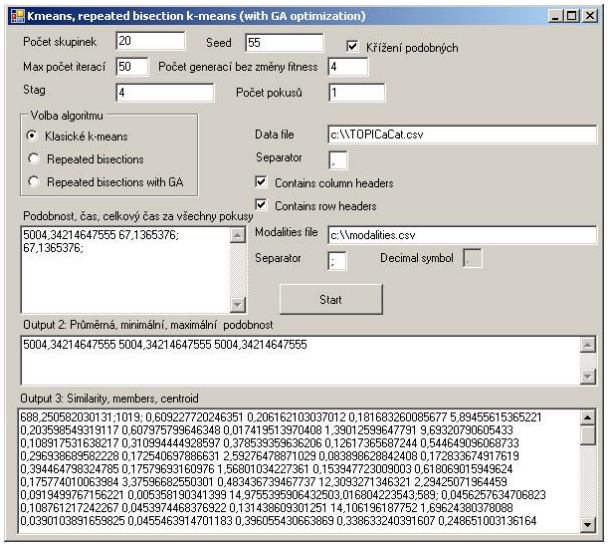


Figure 7.2: Clustering program screenshot

The quality of the clustering was measured using the total similarity computed as a sum of similarities of all the objects with their centroid. The influence of the following algorithm parameters on the quality of clustering was observed:

1. *Spherical K-means*: number of iterations of the algorithm

2. *Repeated Bisection K-means*: number of iterations of the spherical K-means in the bisection step

3. *Repeated Bisection K-means With Genetic Algorithm Initialization*: population size. Common settings: maximum number of epochs 10, mutation parameter $SigFac = 4$. Stopping condition: $Stag = 4$ epochs without fitness change.

There were five runs of each algorithm for one parameter settings to get representative results. Because it is not possible to directly compare the number of iterations for different algorithm, the plots 7.3 contain the number of "evaluations" – the number of times the spherical K-means was run. Spherical K-means is in the heart of the two more complicated algorithms, equations 4.7, 4.8, 4.9 were used to count how many times it was run for the two more complicated algorithms.

Unfortunately, these values are not fully representative because the problem the spherical K-means solves depends in the case of the two more complicated algorithms on the size of the clusters being split. However, this difference should not exceed an order of the magnitude and K-means converges very quickly. As a consequence, this should not cause a large bias.

7.4.1 Algorithm Comparison

The clustering experiment results as depicted on the plots 7.3 did not confirm the hope that using an evolution algorithm to initialize the Repeated Bisection K-means will lead to better results. This can be explained by the fact that K-means converges quickly and although it is

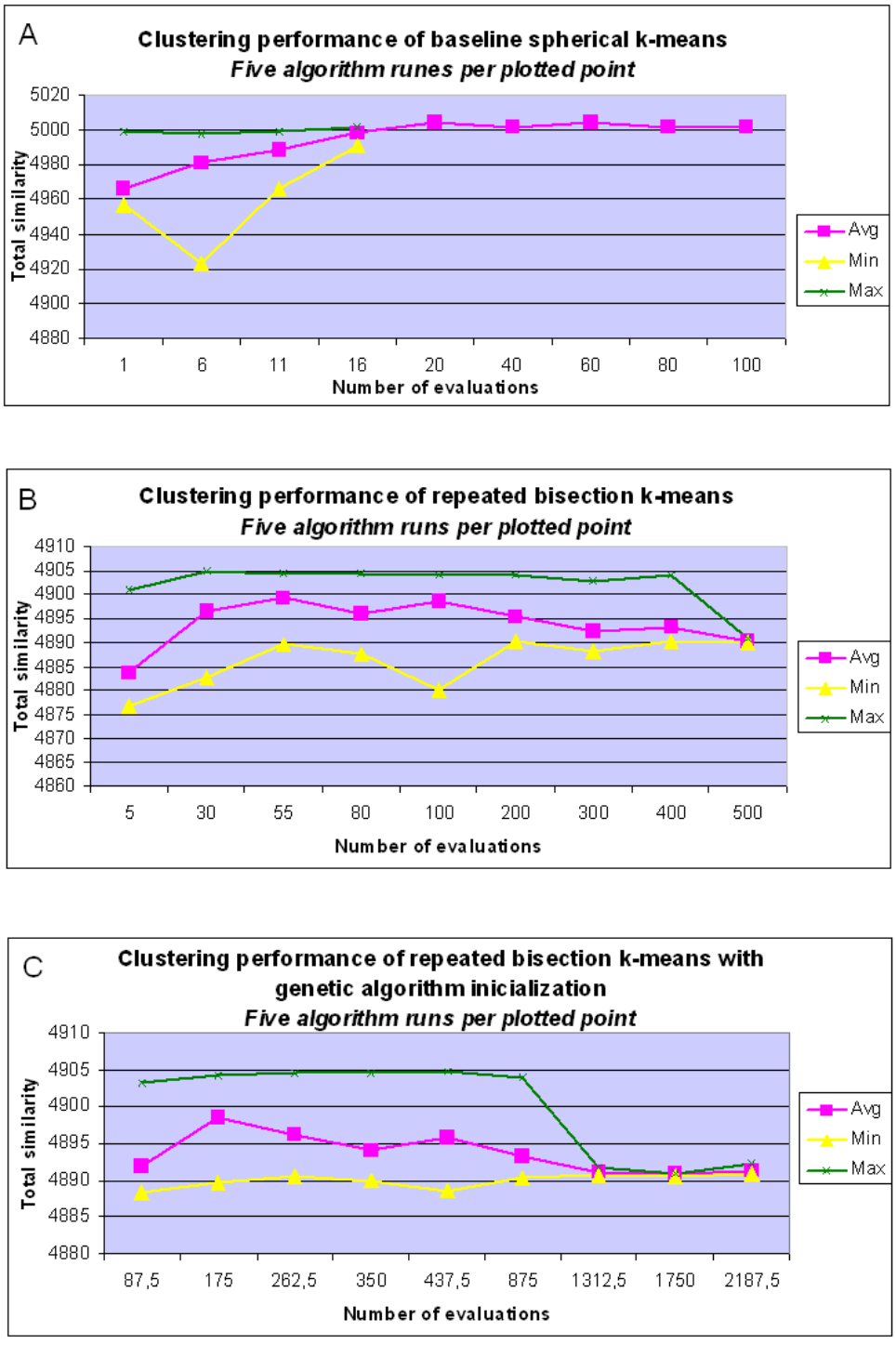


Figure 7.3: The results of the clustering experiments for three algorithms

very sensitive to initialization, only a few its iterations with different random initializations can lead to results which are very difficult to improve.

This tendency can be well seen on the Plot 7.3A of the baseline Spherical K-means. The total similarity of this algorithm virtually stops to increase when the number of algorithm iterations reaches 16. Surprising was the fact that the baseline K-means had a better clustering performance than the both two more complicated algorithms. The possible cause is that a different dataset was used than in the research [25], which recommended repeated bisectional K-means over classical K-means. These researchers used significantly higher dimensionality data, which even contained a modality with weighted keyword vector representation of the web pages.

During the contemplation on the reasons of why the the baseline Spherical K-means surpassed its both more sophisticated hierarchical variants, the following question arose: are repeated bisections the right strategy for data of this character? The number of clusters sought is usually low, so the hierarchical representation looses its importance. Some drawbacks of using a hierarchical clustering may appear, though.

It is not hinted in paper [25] that they would take any advantage of the hierarchical character of the clustering results. In the case study presented in [25], which was conducted on the website `DiamondReview.com`, a dataset of similar number of records as here (3021 user sessions) was used and the number of clusters generated was 6 (same as in this case study).

When trying to find the property of the repeated-bisections clustering which might be responsible for the worse performance, we came up with the situation depicted on Figure 7.4. This figure depicts an artificial dataset, when repeated bisection K-means leads overall to a suboptimal solution, even if every of the steps (bisecting a cluster to two sub-clusters) is optimal[6].

This disadvantage is normally compensated by the fact that resulting clustering is hierarchical and thus can be better visualized.

7.5 Evaluation Of Results

Some of the association rules found with *LISp-Miner* and *Arules* are potentially interesting. However, the rules generated are rather "nuggets" than a comprehensive picture of what is happening on the website.The results of the clustering on the other hand do not explicitly give any guidance on how to increase the ROI of advertisements, but they help to identify important trends and visitor segments.

The association rules discovered provide some clue to the underlying business question "How to increase the ROI of the advertisements?".

The experiment no. 1 (4ft) tells us that visitors coming from the *Seznam* search engine, *Holan* e-shop and *Trekking* e-magazine in the morning hours are likely to request further information. Our advice to the marketing director would be – inspect further possibilities of cooperation with these web sites.

The experiment no. 3 (SD4ft) leads to the hypothesis that last-minute offers on full text search engines have a higher conversion rate in the morning than in the afternoon or in the evening. After the necessary verification, the lesson here for the marketing department could

[6]Repeated Bisection K-means divides cluster on two sub-clusters in a way which (stochastically) minimizes the total similarity of cluster members with the centroids.

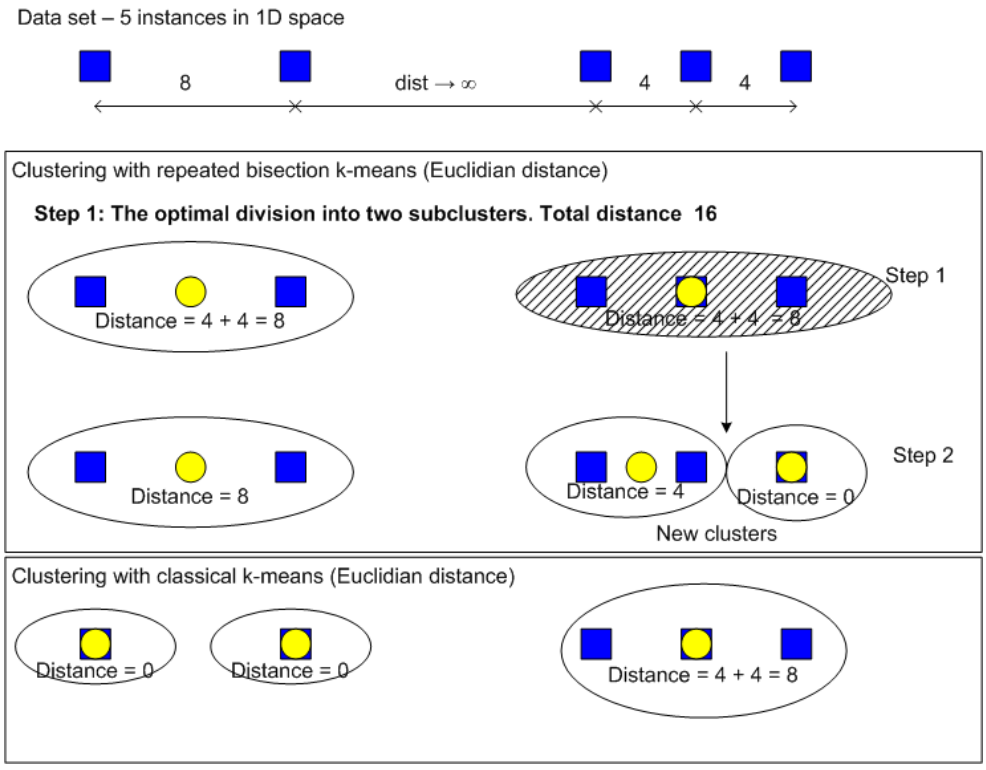


Figure 7.4: Repeated Bisection K-means with is in some circumstances dominated by simple K-means.

#	Similarity	Size	Description
1.	677,520279	788	Hiking
2.	814,6584023	1398	Bulgaria and Monte Negro, Corsica, Cycling, Skiing, Lastminute offers, On-Site search
3.	569,3641568	779	Climbing course, Mountaineering, Extreme Mountaineering, Rafting,
4.	1735,579317	2084	Package holidays, Info, little interest in specific trip offers
5.	696,9996249	999	Package holidays, Trips inclusive, Sightseeing trips with hiking/accommodation, Lipari, high interest in trips
6.	507,5006221	596	Exotic trips, Expeditions

Table 7.5: Clusters created with the baseline (spherical) K-means

be – advertise more heavily last-minute trips on search engines in the morning, and then pull out most of the advertisements for the rest of the day.

From the data mining perspective, it could be interesting to further elaborate the significance of the morning attribute value. Interestingly enough, it appears in both rules highlighted above. For example, it might have gained significance simply because there is more traffic in the morning than later in the day.

Some of the rules that we obtained seem to us subjectively interesting, but their real potential of contributing to increase in the ROI of advertisements would have to be evaluated by a marketing expert. In any case, association rules in principle provide only partial and non-systematic overview of the domain. For more comprehensive analysis, these results were complemented by clustering on using only a subset of attributes.

Presenting the results of clustering is more complicated. The mathematical measures such as total similarity presented in Section 7.4.1 are indicative of the quality of clustering results, but the evaluation should come from the domain expert, who should ultimately use the outcome of the clustering for the advantage of the enterprise.

Cluster centroids were used to represent the results to him. Because a mere displaying of the weights assigned to the individual dimensions would be confusing for the expert, we came up with a representation in Tables 7.5, where each cluster is explained as a visitor segment.

It is described by selected centroid features to denote how the visitors belonging to this cluster segment differ from visitors belonging to other segments. In addition, several other characteristics are used: The intra-cluster similarity represents how homogeneous the cluster is and the number of elements of the cluster indicates what portion of the website visitors might exhibit the browsing patterns described by the cluster.

The user-friendly representation of two of the clustering algorithms is contained in Tables 7.5. Comparing these results is difficult. Nevertheless, the expert subjectively concluded that the results of the baseline spherical K-means are crisper and clearer than the results of Repeated Bisection K-means With Genetic Algorithm Initialization.

The case study also tried to give the answer to which algorithm is most suitable for the given task. For mining associations, the clear winner is the *LISp-Miner* system. It is available free for download from `http://lispminer.vse.cz`. In comparison with R, the *Arules* embodying environment, *LISp-Miner* does not offer such a large suite of in-the-box

#	Eq.	Similarity	Size	Description
1.	4.	1224,875248	1370	General info, little interest in trips
2.	3. , 1.	933,455453288908	1312	Climbing course, Mountaineering, Hiking, High interest in trip offers
3.	2.	413,120198692062	529	Last-minute, Skiing, Search, no interest in trips
4.	2., 5.	1021,89387600393	1492	Bulgaria, Monte Negro, Cycling, Golf,Corsica, Package holidays, Search intensive, Info
5.	2., 6.	569,872519399312	820	Cycling, Exotic trips, Expeditions, Skiing,Rafting, Trips
6.	5.	727,633515497564	1121	Monte Negro, Golf, Corsica, Lipari, Sightseeing trips with hiking, Accommodation, Interest in trips

Table 7.6: Clusters created with repeated bisections K-means with genetic algorithm initialization. The values Eq column indicate to which cluster from Table 7.5 the row most corresponds.

data preprocessing and cleansing tasks. On the other hand, it provides a graphical user interface to the complete data mining process.

We have found the preprocessing and visualization tasks offered by *LISp-Miner* to be sufficient in our web usage mining scenario and ultimately easier to use than the much richer but command-line controlled functionality of *R*. R would probably have an advantage for more complex data mining tasks than presented here because it allows to integrate a range of existing data mining and statistical packages as well as to write ones own code.

An option for the data analyst, who needs a customizable data mining platform and does not want to loose the extras offered by GUHA, *Ferda dataminer* might be an option. *Ferda* is a modern data mining environment, which currently implements several GUHA procedures. It is easily extensible, a range of programming languages is supported including C#, the implementation language of the clustering algorithms presented in this thesis. It remains for further work to adapt some of the algorithms presented here for use with the *Ferda dataminer*.

Regarding the choice of the clustering algorithm, the case study shows that the classical K-means algorithm gives better results for the dataset used. Using more sophisticated clustering algorithms lead to comparable results but with higher computational costs. It may be, however, conceived that Repeated Bisection K-means may perform better on a higher dimensionality dataset. Due to its hierarchical nature, it also brings a potential for visualization, which this case study left unexploited. The Repeated Bisection K-means With Genetic Algorithm Initialization did not perform as well as expected. The suspected reason is the low volume of data. It is a question of further experiments to show whether it can deliver any benefit on a larger dataset.

Conclusion

Web Usage Mining does not have a long history, but the number of scientific papers dealing with this topic is overwhelming. This is curious given the fact that there are not so many applications. The majority of the papers referenced in this thesis are rather theoretical works with limited possibilities of being applied in practice. Real world applications require easily readable comprehensive results and this is far from what most WUM methods deliver. By presenting selected approaches from the available literature this thesis strived to show that data mining based clickstream analysis can already help businesses to gain useful knowledge about their website visitors. A thesis goal was also to design a heuristic that could help to unveil visitors' information needs, which would hopefully contribute to widening the applicability of Web Usage Mining in practice.

The number of publications related to Web Usage Mining is high as is the number of different problems they address. To avoid too shallow coverage of the topics, the author decided to present approaches only to selected problems, but from virtually all phases of the data mining process, and illustrate some of them in a case study.

The remainder of the conclusion is structured as follows. First, there is an overview of the problems covered in this thesis. This overview is loosely structured according to the order in which the problems appear in the phases of the CRISP-DM methodology. Then, it discussed to which extent the thesis met the second goal – proposing a heuristic that would be able to identify the main interest of a visitor.

Finally, proposals for further work are made.

Overview Of Problems Covered

Chapter 1 showed that there is an increasing pressure for WUM methods that could be deployed to answer acute business needs pertaining to the click-fraud and conversion-fraud detection. Chapter 5 demonstrated a specific vulnerability of a widely used clickstream analysis solution to emphasize the need for increased quality of collected data and strong fraud detection mechanisms.

Web Usage Mining is undoubtedly a promising field. Unlike in most other data mining application areas, the quantity of clickstream data is abundant as more people go on-line. What however hampers the application of modern data mining techniques are the high costs of background knowledge acquisition, especially of the semantics. These costs are actually rising as commercial web sites grow both in the number of pages and products offered as well as in the richness of the information presented. Chapter 2 recapitulated what data can be collected and by which means. Chapter 6 presented practical solution how to elegantly bridge the semantic gap using the infrastructure already present on most e-commerce websites.

Chapter 3 showed that data representation is not just a set of routine data transforma-

tions and cleansing tasks, but that it can also be a science in itself. The variable nature of clickstreams forces the data analysts to come up with ingenious solutions based on advanced mathematics and statistics. This chapter also presented a promising classifier-based approach to dimensionality reduction, which unlike some other statistical and mathematical techniques preserves the understandability of the preprocessed data to domain expert.

Chapter 4 gived some details about two association rule mining systems and three clustering algorithms, which might be suitable for Web Usage Mining. One of the three clustering algorithms contains an innovative element – initialization with genetic algorithm. This innovation was hoped to overcome the undesirable property of K-means: clustering results may vary substantially for different selections of the initial cluster centroids.

Chapter 7 put all these algorithms including the newly proposed one to a test. The two data mining systems LISp-Miner and Arules and the three clustering algorithms were compared in a real world data mining scenario. It is concluded that LISp-Miner offers better value both in functionality and scalability. Arules had a slightly better performance. The clustering algorithm of choice on this dataset was Spherical K-means.

Evaluation Of The Proposed Heuristics

The design of the heuristic which would be able to solve the problem *of identification of the main interest of a visitor* proved more complicated than it originally seemed. The theory of the heuristic was, however, finished and it is available in Section 3.7.

In a nutshell, this heuristic can be used for the reduction of dimensionality of the input data. Consider the following common situation. Each page in the visitor clickstream is tagged by a semantic attribute a. This attribute can have a different value for each page in the clickstream. If this attribute is to be used in mining, many algorithms require that the visitor profile, which represents the clickstream of one visitor, contains as many attributes as there are possible values in the original attribute. The proposed heuristic allows to determine the prevalent attribute value for the given visit, which allows the original attribute to be represented only by the prevalent value. The proposed heuristic is a function that scores each of the possible attribute values. The attribute value with the highest score is selected. This function is learned for every website and attribute based on a training set, which comprises of visitor profiles for which the prevalent attribute value is known. Genetic programming is used to learn the function itself, not only its coefficients.

The parameters of the heuristic could not be learned because of the lack of data. The designed heuristic relies on a provision of consistent and quality semantic data (ontology) of the website and on the availability of the training set. Such data are very expensive to obtain manually and automated approaches for extraction of semantics from hypertext are complicated and unreliable. Acquiring the training set is a peculiar problem itself, because it should contain clickstreams with explicitly declared visitor's interest in one of the attribute values.

The author therefore opted to design a new clickstream collection software dubbed Google Analytics INterceptor or GAIN in short. GAIN is presented in Section 6 and among other features it introduces a novel (to the best of the author's knowledge) approach to semi-automatic collection of semantic data. This software acts as a kind of plug-in for Google Analytics and can be seamlessly integrated into existing websites. To the best of the author's knowledge this is the first such plug-in for Google Analytics provided by a third party.

The acquisition of training set is solved by GAIN as well. The visitors of e-commerce

website, who buy some product, explicitly express their interest in the attributes of the product. Such visitors are called converted visitors. If the attribute on which the heuristic is applied is among the attributes of the products sold, then clickstream of converted visitors can be used as the training set.

In Section 3.6 the author also attempted to design, but not test, a *heuristic for determining the relevancy of results returned by an Internet Search engine for a given website*. Several other new attributes are proposed in this section as well. These attributes were designed to enrich the data for association rule mining, which is reportedly the most common Web Usage Mining technique used. The need for this attributes was demonstrated in Section 7.3 which also presents some experiments. The results of these experiments (carried out without the proposed attributes) were not very conclusive. The proposed attributes, if used, could have arguably improved the quality of results.

Because neither of the proposed heuristics was tested in practice, the author decided to implement and test existing Web Usage Mining approach, which would serve a similar purpose. The selected approach was *multi-modal clustering of visitors* according to the topic of pages they visited.

Three clustering algorithms, including the novel one, were run on the same dataset. Surprisingly, the best clustering results and best performance were achieved by the simplest algorithm – Spherical K-means. The reasons are unclear, but there were several hypotheses offered. The relatively low dimensionality of data might have been a factor, because one of the more sophisticated algorithms, Repeated Bisection K-means, was previously found to give better results than simple K-means, but on a higher dimensionality dataset. It was also shown that the unexploited hierarchical nature of both the more sophisticated algorithms may have contributed to the mediocre results as well.

Hopefully, this thesis helped to shed some light on the usability of Web Usage Mining techniques in E-commerce scenarios. The approach of this thesis was comprehensive, all parts of the CRISP-DM cycle were covered, most of them both in theory and practice. A lot of remains for further work though.

Further work

Techniques and approaches presented in this thesis offer a large potential for further work. Web Usage Mining is a young discipline with a lot of unexplored areas. New challenges and opportunities emerge with the advent of Semantic Web and the AJAX driven Web 2.0.

What regards possible *elaboration of problems closely related to thesis goals*, the research pertaining to the proposed heuristics should be further developed in the first place. Only theoretical foundations have been laid down so far. There is not yet any implementation, which would test the feasibility of the heuristic in practice.

The theory itself could be further developed as well. Concerning the identification of the main interest, there is currently only one winning value of each attribute and the other values are disregarded. It is desirable to handle the runner-up attribute values in a more decent way, because they are still a valuable expression of the visitor's preferences.

Particularly, the *Utility additive* (UTA) method [71] for disaggregate analysis of preferences could be used to gain additional insight into what are the visitor preferences. This method is backed by the Multiattribute utility theory.

UTA is an ordinal regression method using linear programming, which takes as input a set of alternatives ordered by the user. Each alternative is described by several attributes.

If a page view is approached as the alternative, its semantic descriptors are the attributes and the Score assigned to the pageview denotes the ordering, then the UTA method could be applicable also on web usage data. This method should be able to determine the weights that the visitor implicitly assigns to the various properties (attributes) of the desired product.

This thesis is entitled Clickstream analysis, although it largely deals with experimental approaches, to "hint how the clickstream analysis routine could be redesigned with the use of advanced algorithms from the Web Usage Mining toolbox". To contribute to this vision, almost all the algorithms, which were implemented in this thesis, were programmed in languages supported by the Ferda Dataminer to make their possible inclusion into this environment easier. Ferda Dataminer [5] is a new extensible visual data mining environment, whose traits would make it the ideal common platform for open source WUM preprocessing and data mining tasks.

Bibliography

[1] Customize google. URL `http://customizegoogle.blogspot.com/`. [Online; accessed 30-July-2007].

[2] Lecture 5: Dimensionality reduction (pca). URL `http://courses.cs.tamu.edu/rgutier/cs790_w02/l5.pdf`. University Course material.

[3] *LISp-Miner*. URL `http://lispminer.vse.cz`. [1.1.2006].

[4] Webtrends advises sites to move to first-party cookies based on four-fold increase in third-party cookie rejection rates. 2005. URL `http://www.webtrends.com/AboutWebTrends/NewsRoom/NewsRoomArchive/2005/CookieRejection.aspx`. [1.1.2006].

[5] Ferda, nov vizuáln prosted pro dobván znalost. 2006.

[6] Why not google web analytics?, 2007. URL `http://www.webtrafficiq.com/home/white_paper_google_web_analytics.pdf`. White paper.

[7] A. Abraham and V. Ramos. Web usage mining using artificial ant colony clustering and genetic programming, 2003.

[8] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994. ISBN 1-55860-153-8.

[9] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In Philip S. Yu and Arbee S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[10] Hisham Al-Mubaid and Hoa A. Nguyen. Semantic distance of concepts within a unified framework in the biomedical domain. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 142–143, New York, NY, USA, 2007. ACM Press. ISBN 1-59593-480-4. doi: http://doi.acm.org/10.1145/1244002.1244037.

[11] Charles Annis. Statistical engineering. URL `http://www.statisticalengineering.com/curse_of_dimensionality.htm`. Web article.

[12] Vinod Anupam, Alain Mayer, Kobbi Nissim, Benny Pinkas, and Michael K. Reiter. On the security of pay-per-click and other Web advertising schemes. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1091–1100, 1999.

[13] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 144–153, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-340-9. doi: http://doi.acm.org/10.1145/1137856.1137880.

[14] Daniel Ashlock. *Evolutionary Computation for Modeling and Optimization*. Springer, 2005. ISBN 978-0387221960.

[15] Martin Atzmuller and Frank Puppe. A methodological view on knowledge-intensive subgroup discovery. In Steffen Staab and Vojtech Svtek, editors, *Proceedings of EKAW 2006 - 15th International Conference on Knowledge Engineering and Knowledge Management Managing Knowledge in a World of Networks*, volume 4248 of *Lecture Notes in Computer Science*, pages 318–325. Springer.

[16] Michal Barla. Interception of user's interests on the web. In Vincent P. Wade, Helen Ashman, and Barry Smyth, editors, *AH*, volume 4018 of *Lecture Notes in Computer Science*, pages 435–439. Springer, 2006. ISBN 3-540-34696-1.

[17] Michal Barla. *Zachytenie záujmov pouvatela na webe*. Fakulta informatiky A informačných Techonológií - Slovenská technická univerzita v Bratislave, 2006. Diplomová práce.

[18] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-391-X. doi: http://doi.acm.org/10.1145/502512.502546.

[19] Christian Borgelt. *Apriori documentation*. URL `http://fuzzy.cs.uni-magdeburg.de/~borgelt/doc/apriori/apriori.html`. [4.5.2007].

[20] Christian Borgelt and Rudolf Kruse. Induction of association rules: A priori implementation, 2002.

[21] S. C. Bradford. Sources of information on specific subjects. *J. Inf. Sci.*, 10(4):173–180, 1985. ISSN 0165-5515.

[22] Jirachai Buddhakulsomsiri. Sequential pattern analysis for automotive warranty data mining, May 2006. URL `http://www.engin.umd.umich.edu/hpceep/tech_day/download.php?year=2006&dept=IMS&target=2006_IMS_slideshow_pdf_00296.pdf`.

[23] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Vips: a visionbased page segmentation algorithm, 2003.

[24] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rdiger Wirth. *CRISP-DM 1.0 Step-by-step Data Mining Guide*, June 2000. URL `http://www.crisp-dm.org/CRISPWP-0800.pdf`. [1.1.2006].

[25] Ed H. Chi, Adam Rosien, and Jeffrey Heer. Lumberjack: Intelligent discovery and analysis of web user traffic composition. In *Proc. of ACMSIGKDD Workshop on Web Mining for Usage Patterns and User Profiles*. ACM Press, 2002.

[26] Lotesoft co. Clicking agent. URL `http://www.clickingagent.com/softcaca.html`. [8.5.2007].

[27] ComScore. Cookie-based counting overstates size of web site audiences. URL `http://www.comscore.com/press/release.asp?press=1389`. Press Release. April 16, 2007.

[28] Robert Cooley. The use of web structure and content to identify subjectively interesting web usage patterns. *ACM Trans. Inter. Tech.*, 3(2):93–116, 2003. ISSN 1533-5399.

[29] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Web mining: Information and pattern discovery on the world wide web, 1997.

[30] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[31] Miguel Gomes da Costa Junior and Zhiguo Gong. Web structure mining: an introduction. In *Proceeding of the 2005 IEEE International Conference on Information Acquisition*, June 2005. ISBN 0-7803-9303-1.

[32] Michael Etgen and Judy Cantor. What does getting wet (web event-logging tool) mean for web usability?, 1999.

[33] Federico Michele Facca and Pier Luca Lanzi. Mining interesting knowledge from weblogs: a survey. *Data & Knowledge Engineering*, 53(3):225–241, 2005. ISSN 0169-023X. doi: http://dx.doi.org/10.1016/j.datak.2004.08.001.

[34] Peter Fletcher, Alex Poon, Ben Pearce, and Peter Comber. *Practical Web Traffic Analysis: Standards, Privacy, Techniques, Results.* Glasshaus, Birmingham, 2002. ISBN 1-903151-18-3.

[35] Imola Fodor. A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Lab., CA (US), 2002. URL `http://www.osti.gov/energycitations/product.biblio.jsp?osti_id=15002155`.

[36] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases - an overview. *Ai Magazine*, 13:57–70, 1992.

[37] Marina L. Gavrilova, Osvaldo Gervasi, Vipin Kumar, Chih Jeng Kenneth Tan, David Taniar, Antonio Laganà, Youngsong Mun, and Hyunseung Choo, editors. *Computational Science and Its Applications - ICCSA 2006, International Conference, Glasgow, UK, May 8-11, 2006, Proceedings, Part V*, volume 3984 of *Lecture Notes in Computer Science*, 2006. Springer. ISBN 3-540-34079-3.

[38] Bart Goethals. Fimi'03 – frequent itemset mining implementations. URL `http://fimi.cs.helsinki.fi/`.

[39] Antone Gonsalves. Google gets closer to firefox, October 2005. URL `http://www.informationweek.com/showArticle.jhtml?articleID=173601284`.

[40] Joshua Goodman. Pay-per-percentage of impressions: An advertising method that is highly robust to fraud. In *Presented at the ACM E-Commerce Workshop on Sponsored Search Auctions*, 2005.

[41] Ted Goranson. The agile virtual enterprise: Cases, metrics, tools. URL `http://www.sirius-beta.com/ALICEsupp/FarOutIdeas/conceptlattices.html`.

[42] Laura A. Granka, Thorsten Joachims, and Geri Gay. Eye-tracking analysis of user behavior in www search. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 478–479, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-881-4. doi: http://doi.acm.org/10.1145/1008992.1009079.

[43] Michael Hahsler, Bettina Grun, and Kurt Hornik. *Introduction to arules Mining Association Rules and Frequent Item Sets*. URL `http://cran.mirroring.de/doc/vignettes/arules/arules.pdf`. [4.5.2007].

[44] Petr Hájek and Tomáš Havránek. *Mechanizing Hypothesis Formation*. Springer-Verlag, 1978. ISBN 3-540-08738-9.

[45] Petr Hájek, Martin Holena, and Jan Rauch. The guha method and foundations of (relational) data mining. In Harrie C. M. de Swart, Ewa Orlowska, Gunther Schmidt, and Marc Roubens, editors, *Theory and Applications of Relational Structures as Knowledge Instruments*, volume 2929 of *Lecture Notes in Computer Science*, pages 17–37. Springer, 2003. ISBN 3-540-20780-5.

[46] David J. Hand, Padhraic Smyth, and Heikki Mannila. *Principles of data mining*. MIT Press, Cambridge, MA, USA, 2001. ISBN 0-262-08290-X.

[47] Markus Hegland. Algorithms for association rules. *Advanced lectures on machine learning*, pages 226–234, 2003.

[48] N. Immorlica, K. Jain, M. Mahdian, and K. Talwar. Click fraud resistant methods for learning click-through rates. In *Presented at the Workshop on Internet and Network Economics (WINE)*, 2005.

[49] Il Kim, Bong Joon Choi, and Kyoo Seok Park. A study on web-usage mining control system of using page scroll. In Sven F. Crone, Stefan Lessmann, and Robert Stahlbock, editors, *DMIN*, pages 337–342. CSREA Press, 2006. ISBN 1-60132-004-3.

[50] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[51] Tomáš Kliegr. Clickstream analysis - the semantic approach. In Helena Sofia Pinto and Martin Labsky, editors, *Poster and Demo Proceedings of EKAW 2006 - 15th International Conference on Knowledge Engineering and Knowledge Management Managing Knowledge in a World of Networks*, 2006. ISBN 80-86742-15-6.

[52] Tomáš Kliegr. Příprava dat pro clickstream analýzu. In Jén Paralič, Jiří Dvorský, and Michal Krátký, editors, *Sborník konference Znalosti 2006*. VŠB-TU Ostrava, 2006. ISBN 80-248-1001-8.

[53] Tomáš Kliegr. *Search Engine Optimization And Web Metrics*. University of Economics in Prague, Prague, 2006. Bachelor Thesis [In Czech].

[54] Tomáš Kliegr. Mining conversion patterns from click streams. In Mikulecký Peter, Jiří Dvorský, and Michal Krátký, editors, *Sborník konference Znalosti 2007*. VŠB-TU Ostrava, 2007. ISBN 978-80-248-1279-3.

[55] Milos Kovacevic, Michelangelo Diligenti, Marco Gori, Marco Maggini, and Veljko Milutinovic. Recognition of common areas in a web page using visual information: a possible application in a page classification, 2002.

[56] Vladimír Kvasňička, Jiří Pospíchal, and Peter Tiňo. *Evolučné algoritmy*. STU Bratislava, 2000. ISBN 80-227-1377-5.

[57] A. Metwally, D. Agrawal, and A. Abbadi. Using association rules for fraud detection in web advertising networks, 2005.

[58] Elinor Mills. Bumpy start for google analytics giveaway. *CNET news.com*, 2005. URL `http://news.com.com/Bumpy+start+for+Google+analytics+giveaway/2100-1032_3-5956308.html`. Published: November 16, 2005.

[59] Bamshad Mobasher and Honghua Dai. *Integrating Semantic Knowledge with Web Usage Mining for Personalization*, chapter XIII, pages 276–306. In , Scime [69], 2004. ISBN 1591404150.

[60] I.C. Mogotsi. *What did they cover?* University of Stellenbosch, 2006. Disertan práce.

[61] Laila Paganelli and Fabio Paternò. Intelligent analysis of user interactions with web applications. In *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, pages 111–118, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-459-2. doi: http://doi.acm.org/10.1145/502716.502735.

[62] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[63] J. M. Pena, J. A. Lozano, and P. Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recogn. Lett.*, 20(10):1027–1040, 1999. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/S0167-8655(99)00069-0.

[64] Gregory Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*, pages 229–248. AAAI/MIT Press, 1991. ISBN 0-262-62080-4.

[65] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[66] Jan Rauch and Milan Simunek. Guha method and granular computing. In Xiaohua Hu, Qing Liu, Andrzej Skowron, Tsau Young Lin, Ronald R. Yager, and Bo Zhang, editors, *GrC*, pages 630–635. IEEE, 2005.

[67] Michael K. Reiter, Vinod Anupam, and Alain Mayer. Detecting hit shaving in click-through payment schemes. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, pages 155–166, 1998.

[68] Gerard Salton. *Introduction to Modern Information Retrieval (McGraw-Hill Computer Science Series)*. McGraw-Hill Companies, September 1983. ISBN 0070544840.

[69] Anthony Scime. *Web Mining: Applications and Techniques*. IGI Publishing, Hershey, PA, USA, 2004. ISBN 1591404150.

[70] Cyrus Shahabi, Adil Faisal, Farnoush Banaei Kashani, and Jabed Faruque. INSITE: A tool for interpreting users? interaction with a web space. In *The VLDB Journal*, pages 635–638, 2000.

[71] Yannis Siskos. *UTA Methods*, chapter XVIII, pages 297–334. Springer. ISBN 978-0-387-23081-8.

[72] StatSoft. *StatSoft Electronic Textbook*. URL `http://www.statsoft.com/textbook/glosc.html`. Available online.

[73] Jim Sterne. *Web Metrics: Proven Methods for Measuring Web Site Success*. Wiley, New York, 2002. ISBN 0-471-22072-8.

[74] Achilleas Stogioglou Steve McLaughlin and Justin Fackrell. Introducing higher order statistics (hos) for the detection of nonlinearities. URL `http://www.maths.leeds.ac.uk/applied/news.dir/issue2/hos_intro.html`.

[75] Alexander Strehl. *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. The University of Texas at Austin, 2002. PhD Dissertation.

[76] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on web-page clustering. In *Proceedings of the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search (AAAI 2000), 30-31 July 2000, Austin, Texas, USA*, pages 58–64. AAAI, July 2000.

[77] Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Fast computation of concept lattices using data mining techniques. In *Knowledge Representation Meets Databases*, pages 129–139, 2000.

[78] P. Tan and V. Kumar. Discovery of web robot sessions based on their navigational patterns. *Data Mining and Knowledge Discovery*, 6:9–35, 2002.

[79] P. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the Eight A CM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 183, July 2002.

[80] Richard Thomas, Gregor Kennedy, Steve Draper, Rebecca Mancy, Murray Crease, Huw Evans, and Phil Gray. Generic usage monitoring of programming students, 2003.

[81] Alexander Thuzhilin, 2006. URL `http://googleblog.blogspot.com/pdf/Tuzhilin_Report.pdf`. The Lanes Gifts v. Google Report.

[82] Mariângela Vanzin and Karin Becker. Ontology-based rummaging mechanisms for the interpretation of web usage patterns. In Markus Ackermann, Bettina Berendt, Marko Grobelnik, Andreas Hotho, Dunja Mladenic, Giovanni Semeraro, Myra Spiliopoulou,

Gerd Stumme, Vojtech Svátek, and Maarten van Someren, editors, *EWMF/KDO*, volume 4289 of *Lecture Notes in Computer Science*, pages 180–195. Springer, 2005. ISBN 3-540-47697-0.

[83] N. Vidyasagar. India's secret army of online ad 'clickers'. *Times of India*, 2004. URL `http://timesofindia.indiatimes.com/articleshow/msid-654822,curpg-1.cms`. [3.5.2004].

[84] Lukáš Vlček. A guha style association rule mining using prefix tree data structure. In Jén Paralič, Jiří Dvorský, and Michal Krátký, editors, *Sborník konference Znalosti 2006*. VŠB-TU Ostrava, 2006. ISBN 80-248-1001-8.

[85] Wikipedia. Ontology (computer science) - wikipedia the free encyclopedia, 2007. URL `http://en.wikipedia.org/w/index.php?title=Ontology_%28computer_science%29&oldid=144091862`. [Online; accessed 19-July-2007].

[86] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005. ISBN 0120884070.

[87] Yew Kwong Woon, Wee Keong Ng, and Ee-Peng Lim. Evaluating web access log mining algorithms: A cognitive approach. In Bo Huang, Tok Wang Ling, Mukesh K. Mohania, Wee Keong Ng, Ji-Rong Wen, and Shyam K. Gupta, editors, *WISE Workshops*, pages 217–222. IEEE Computer Society, 2002. ISBN 0-7695-1813-3.

[88] Shi Zhong. Efficient online spherical k-means clustering. In *Proceedings of the Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, volume 5, pages 3180–3185. IEEE, July 2000.

Appendix – Plots Of Attributes Used In The Case Study

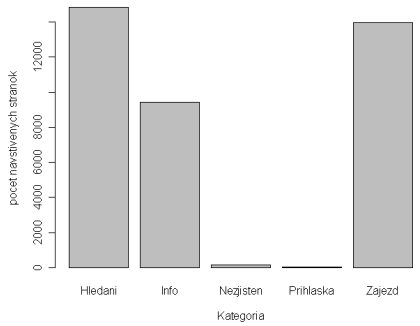


Figure 7.5: CatName Attribute Plot

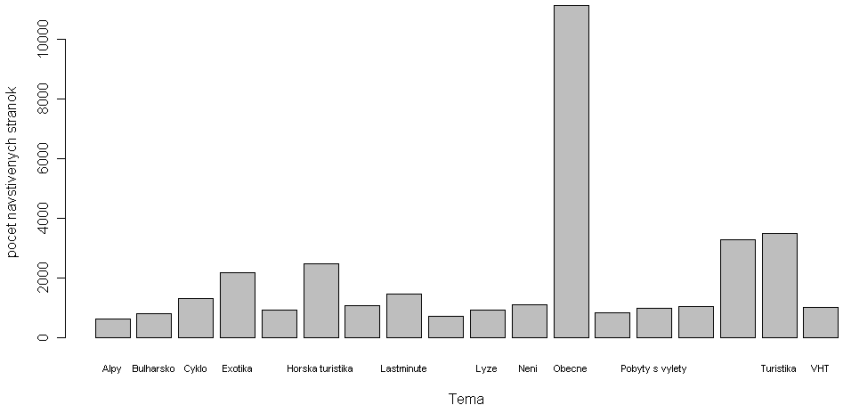


Figure 7.6: Topic Attribute Plot

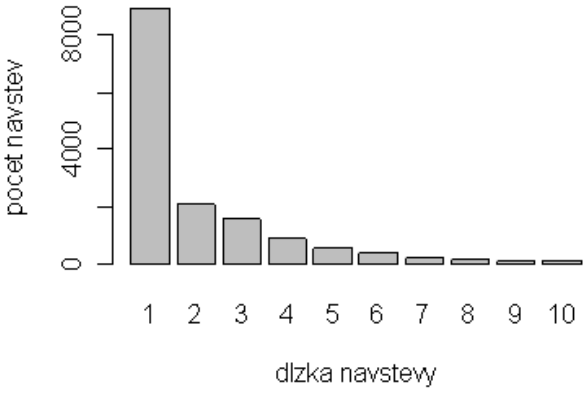


Figure 7.7: Visit Length Attribute Plot

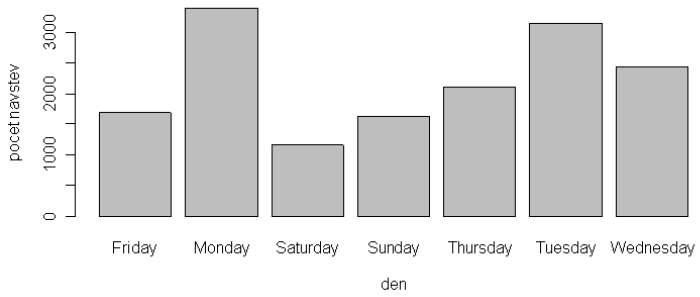


Figure 7.8: Day Attribute Plot

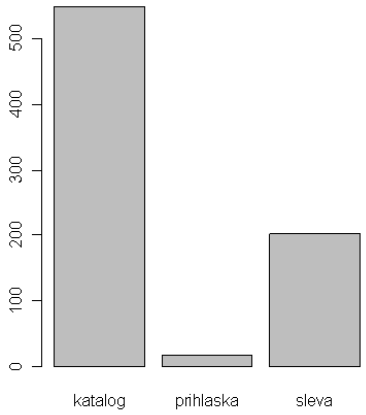


Figure 7.9: Conversion Rate Attribute Plot

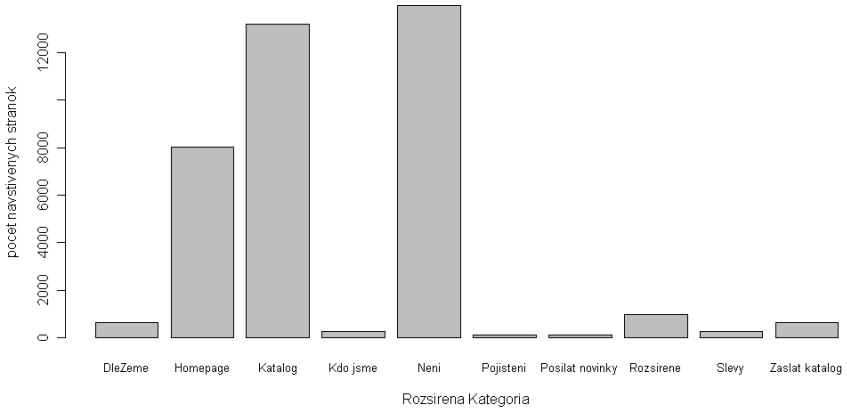


Figure 7.10: ExtCatName Attribute Plot

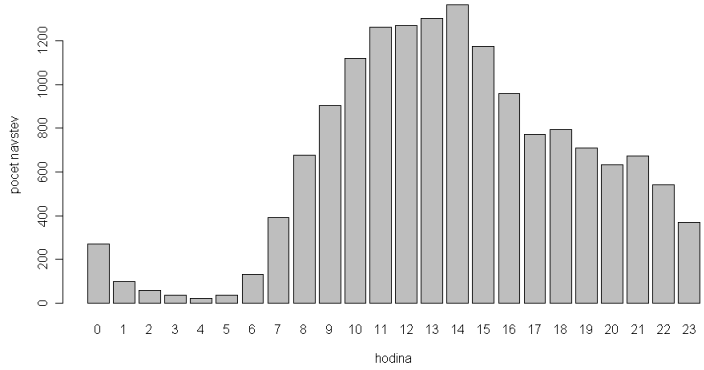


Figure 7.11: Hour Attribute Plot

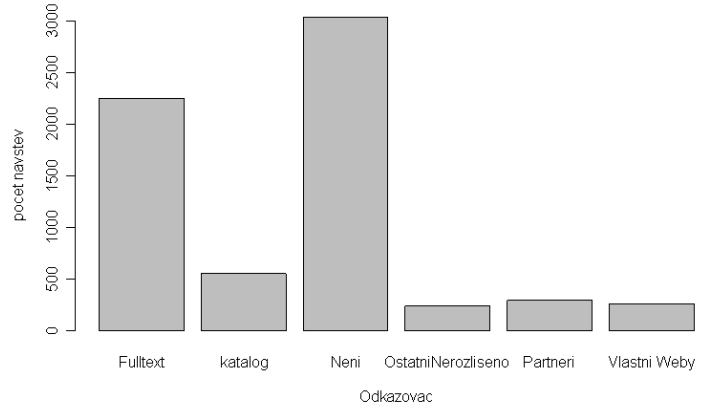


Figure 7.12: Referrer Attribute Plot