Université de Montréal

“WARES”, a Web Analytics Recommender System

par Sedliar Kostiantyn

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la faculté des arts et des sciences
en vue de l’obtention du grade de Maitrise
en Sciences et Informatique

Octobre 2017

Résumé

Il est difficile d'imaginer des entreprises modernes sans analyse, c'est une tendance dans les entreprises modernes, même les petites entreprises et les entrepreneurs individuels commencent à utiliser des outils d'analyse d'une manière ou d'une autre pour leur entreprise. Pas étonnant qu'il existe un grand nombre d'outils différents pour les différents domaines, ils varient dans le but de simples statistiques d'amis et de visites pour votre page Facebook à grands et sophistiqués dans le cas des systèmes conçus pour les grandes entreprises, ils pourraient être shareware ou payés. Parfois, vous devez passer une formation spéciale, être un spécialiste certifiés, ou même avoir un diplôme afin d'être en mesure d'utiliser l'outil d'analyse. D'autres outils offrent une interface d'utilisateur simple, avec des tableaux de bord, pour satisfaire leur compréhension d'information pour tous ceux qui les ont vus pour la première fois. Ce travail sera consacré aux outils d'analyse Web. Quoi qu'il en soit pour tous ceux qui pensent à utiliser l'analyse pour ses propres besoins se pose une question: "quel outil doit je utiliser, qui convient à mes besoins, et comment payer moins et obtenir un gain maximum". Dans ce travail je vais essayer de donner une réponse sur cette question en proposant le système de recommandation pour les outils analytiques web –WARES, qui aideront l'utilisateur avec cette tâche "simple".

Le système WARES utilise l'approche hybride, mais surtout, utilise des techniques basées sur le contenu pour faire des suggestions. Le système utilise certains ratings initiaux faites par utilisateur, comme entrée, pour résoudre le problème du "démarrage à froid", offrant la meilleure solution possible en fonction des besoins des utilisateurs. Le besoin de consultations coûteuses avec des experts ou de passer beaucoup d'heures sur Internet, en essayant de trouver le bon outil. Le système lui–même devrait effectuer une recherche en ligne en utilisant certaines données préalablement mises en cache dans la base de données hors ligne, représentée comme une ontologie d'outils analytiques web existants extraits lors de la recherche en ligne précédente.
Mots–clés: Analytique, système de recommandation, moteur de recherche, ontologies.

Abstract

It is hard to imagine modern business without analytics; it is a trend in modern business, even small companies and individual entrepreneurs start using analytics tools, in one way or another, for their business. Not surprising that there exist many different tools for different domains, they vary in purpose from simple friends and visits statistic for your Facebook page, to big and sophisticated systems designed for the big corporations, they could be free or paid. Sometimes you need to pass special training, be a certified specialist, or even have a degree to be able to use analytics tool, other tools offers simple user interface with dashboards for easy understanding and availability for everyone who saw them for the first time. Anyway, for everyone who is thinking about using analytics for his/her own needs stands a question: "what tool should I use, which one suits my needs and how to pay less and get maximum gain". In this work, I will try to give an answer to this question by proposing a recommender tool, which will help the user with this "simple task". This paper is devoted to the creation of WARES, as reduction from Web Analytics REcommender System. Proposed recommender system uses hybrid approach, but mostly, utilize content–based techniques for making suggestions, while using some user's ratings as an input for "cold start" search. System produces recommendations depending on user's needs, also allowing quick adjustments in selection without need of expensive consultations with experts or spending lots of hours for Internet search, trying to find out the right tool. The system itself should perform as an online search using some pre–cached data in offline database, represented as an ontology of existing web analytics tools, extracted during the previous online search.

Keywords: Web analytics, recommender system, search engine, ontologies.

Table of content

List of Tables

List of figures

List of acronyms

CBF – Content–based Filtering
CF – Collaborative Filtering
CPU – Central Processing Unit
DNS – Dynamic Names System
EU – European Union
GA – Google Analytics
GPL – General Public License
HTTP – Hypertext Transfer Protocol
IMDB – Internet Movie Database, http://www.imdb.com/
IT – Information Technologies
KPI – Key Performance Indicator
MAE – Mean Absolute Error
OS – Operating System
OWL – Web Ontology Language
PC – Personal Computer
PDF – Portable Data Format by Adobe^{tm} corporation
PHP – Personal Home Page scripting language
pLSA – probabilistic Latent Semantic Analysis
REST – Representational State Transfer
RMSE – Root Mean Square Error
ROI – Return of Investment
RS – Recommender System
SVD – Singular Vector Decomposition
TCP – Transfer Control Protocol
TF–IDF – Term Frequency – Inverse Document Frequency
URL – Uniform Resource Locator
W3C – Word Wide Web Consortium
WWW – World Wide Web
XML – Extended Markup Language
YM – Yandex Metrica

Acknowledgments

First and above all, I would like to thank my supervisor, Ph. D. professor Esma Aïmeur of the Department of Computer Science and Operational Research at University of Montreal. Without this woman, this thesis would have been impossible to accomplish, not only because of her help and guidance, but also because of her fate in me. I took a very slow start, I changed the topic, I made mistakes, I had psychological problems with concentration. However, she never gave up on me, she was always friendly during our weekly "team" presentations. Friendly and at the same time strict and serious. I would like to thank her from all my heart!

In addition, I would like to thank Zakaria Sahnoune, who is a Ph. D. student with Esma. He was kind to me and helped several times with advices on my topic, and gave me some general, but useful tips on how to structure and formalize my thesis.

Also I would like to thank Mouna Selmi, she defended her Ph. D. with Esma just before me, and she also helped me several times with useful tips on my topic.

Introduction

In this thesis, I will present a recommender system for making suggestions about the Web Analytics tools, which I called WARES, short from **W**eb **A**nalytics **RE**commender **S**ystem. This system intends to help its users in the selection of "appropriate" web analytics tools, for whatever their personal criteria are.

Problematic

Many problems in the field of recommender systems are already solved, but for particular recommender systems, not in the whole field of recommender systems e.g. in movies, recommendations exist IMDB. In sales exist Amazon and eBay, this system works pretty well, because they already have databases in their specific areas, this data allows them to make recommendations using content–based approach, and they also have many users, which simultaneously permits making suggestions based on collaborative filtering approach. In this paper we will consider the design process of the recommender system, which does not have lots of users. The WARES is intended for a single user, who needs particular web analytics tool, which satisfies his/her particular needs. In this case the best solution for the WARES will be an implementation of the content–based filtering approach. During the development of the WARES we also faced the main issue for the most recommender systems – the "cold start problem" (when recommender system cannot produce recommendations for users or items about which it has not yet gathered "enough" information). To make reliable recommendations the system should have a lot of ratings data for items, but this data comes from a large number of users, which we do not have, which in its turn does not allow us to implement collaborative filtering approach. Another existing problem is constantly changing source data, meaning that to produce a relevant recommendation the system needs "fresh" data, because nobody wants to see irrelevant "expired" recommendations, that is why the proposed recommender system should be, in some way, always aware about web analytics tools on the market. That is why was considered to use an online search, also was considered that WARES should have an ontology for storing data about existing web analytics tools and constantly update this ontology between

user's requests. There is also a flexibility problem, because users have a trend always changing their own preferences, that is why WARES should somehow envisage this, allowing user to change search criteria. An algorithm for making suggestions is needed. A very urgent problem in the Internet space to date is privacy. In WARES privacy problem is solved, because the system does not store any personal data about active user, there is no user database with real names or credit card numbers, e.g. like in Netflix or eBay. Because in the proposed recommender system exists only one abstract user, without any personal information, all the data about web analytics tools, which was found during the recommendation process and some data about user's preferences are stored in the ontology, on user's computer.

Contributions

WARES share many features with existing recommender systems, however WARES adds some novelty in an existing variety of recommender systems. First, it is unique on its own, because it exists several recommender systems about music, movies, consumer goods, however, to the best of my knowledge, there is no such a system for the web analytics. This paper also will present the brand new ontology developed specifically for the domain of web analytics, as well as an algorithm for making recommendations and the algorithm will share adopted features from existing algorithms, in the field of recommender systems.

Plan of this thesis

This thesis is structured as follows: in the first chapter, will be defined basic concepts of the web analytics, and basic notions about recommender system, examples and a short comparison of some most popular existing web analytics tools on the market. In the second we will consider and compare collaborative and content–based algorithms and methods, and possible ways to use these techniques into the WARES. The third chapter describes the structure of the WARES: user's model, application model, data model, ontology, user's interface, and briefly describes how WARES works altogether. The fourth chapter is the validation and a brief comparison of the WARES to some well-known recommender systems on the market. The fifth chapter will conclude this paper, summarizing what was done and possible future works.

Chapter 1 – State of the art

In this chapter, will be describe the domain of "Web Analytics", what is it and what it consist of. Before starting developing WARES recommender system, a preliminary study about recommender system as a notion should be done, to understand it. Basics will be explained about recommender systems, how they works in general and what tasks they pursued, different approaches to the solution of the recommendations problem. Finally, in the end of this chapter examples of existing web analytics tools will be shown, with a summarizing table comparing their important features.

1.1 – Web Analytics

"Web analytics" is not a simple discipline, it is a complex of technologies and methods allowing collection, measurement, analysis and reporting of websites and web application data in order to understand, improve and optimize their performance [Zheng and Peltzverger, 2015]. Today, web analytics is used for different purposes, including marketing, traffic monitoring, information architecture, e–commerce optimization, advertising, web development, web–based campaigns, website performance improvement, etc. Here are some fundamental areas where web analytics is used:

- **Improving performance and identification problems in web applications**: web page loading metrics such as average *page load time* by the browser and *geographic location* are used for measuring website's performance. Analysis of the real-time load allows detecting and investigate problems concerning website's performance, e.g. by optimizing the size of downloadable images or modifying HTTP headers used for caching of the website content. Web analytics metrics, e.g. *click path*, might also help to detect website errors, such as user's clicks on links leading to incorrect or "blank" URL. For the developers of the web applications, web analytics could be applied for detection of the code errors connected with website interactions.

- **Improving website (application) design and user experience**: by modifying existing website appearance, e.g. changing the order of presented content and its visual representation to the end-users, thus improving navigation convenience and layout, or by analyzing clickstream information predicting the "best" positions for ads. banners. Using web analytics feature called *"heat map"* could help to identify user's degree of interest and attention to the certain areas of the website, which is again very important while placing ad banners or key buttons like "buy it now", "sign in", "register", etc.
- **Tracking and measuring success of actions**: activities such as online *commercial campaigns,* and *polls* need to be somehow measured, web analytics helps to cope with this by tracking a wide variety of *traffic sources*, *marketing channels*, and *visitor types*. In these campaigns, an important common question is "how and where participants found that information?" Common traffic source metrics used in web analytics allows us to answer this question very precise, by tracking direct traffic from user's emails, browser, social media and mobile devices, if not used anonymizers like Tor browser or multiple virtual private networks.
- **Optimizing e–Commerce and improving e–CRM (customer relationship management):** by analyzing different data about previous customer's interactions with website thus improving business relationships with customers, targeting on customer retention and ultimately driving sales growth. Web analytics greatly helps to analyze usage of website date and content, allowing achieving different goals, e.g. increase spending time in social networks by reducing "bounce rate" of users, increase traffic capacity through certain websites for achieving maximum revenue from advertisements, or improve sales increase by helping to detect the most popular products.

Historically, web analytics techniques usually are separated into two major categories: *on–site* and *off–site* web analytics. On–site web analytics refers to data collected on the current site, it is used to effectively measure many aspects of direct user–website interactions, such as *number of visits*, *click path*, *time on site*, etc. Off–site analytics are usually offered by third party companies with using external web log storages, e.g. on the side of service provider or any other host. However, this meaning is quite blurred, mainly because of vendors providing tools that

span both categories. Many different vendors, e.g. Google Corporation with its Google Analytics, provide on–site web analytics software and services.

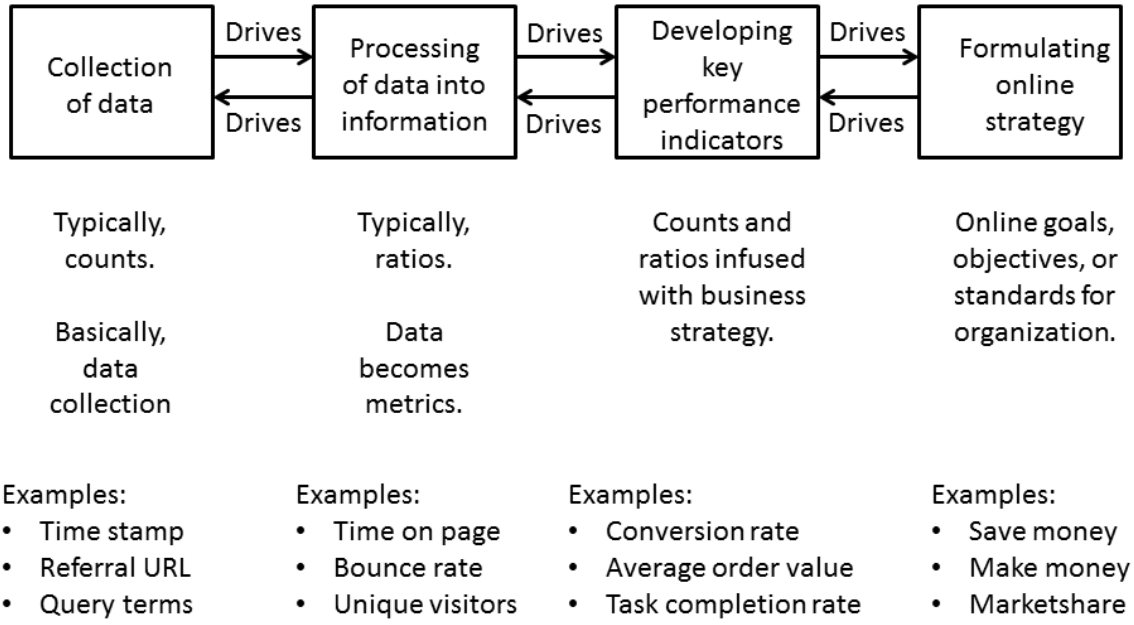


Figure 1: Basic steps of Web Analytics [Clifton, 2010]

It exists many ways to implement web analytics process, but in common sense, they are all the same sequence of actions, typical basic steps in web analytics is shown in figure 1. [Clifton, 2010] let us consider those four essential steps:

- **Collection of data**: collection of the basic elementary data, usually, this data is *counts* of some things e.g. *visits, clicks*. The objective of this stage is just to gather *raw data* for further processing.
- **Processing of data into information**: on this stage, usually making *counts* and makes *ratios*. The objective of this stage is to take the data and transform it into information, which is in most cases called "*metrics*".
- **Developing of KPIs**: this stage focuses on using collected ratios and counts, and express them as a business solutions and strategies, referred to as Key Performance Indicators or KPIs.

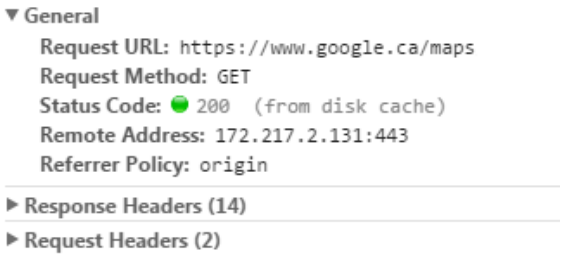
- **Formulating online strategy**: reaching business effectiveness in online goals, objectives, and standards for the organization or business, these strategies usually related to making money, saving money, or increasing market share.

Each stage impacts on the following stage or can affect on the preceding stage or follow it. For example, sometimes specificity of the data available for collection affects the online strategy and the online strategy affects the data to be collected.

1.1.1 – Web Analytics data sources

Data for web analysis can be gathered from *surveys, market reports, competitor comparison, public information*, etc. but mainly the data comes from four data sources [Hu and Cercone, 2004] categorized into following types:

- **Direct HTTP data traffic:** is the user browser passing a web session or visit, which is started without a referer (HTTP header field that identifies the address of the web page). By checking the referrer we can see where the request is originated. An example of the typical HTTP request message is shown in figure 2. An HTTP request consists of a request command shown in the first line and HTTP headers.
- **Application level data sent with HTTP requests:** data generated and processed by application-level programs, such as JavaScript, PHP, and ASP.Net, it is usually embedded into HTTP requests. It includes "*Session*" data, which identifies client interactions with the website. Session data usually sent as URL parameters or session cookies. They are very important for calculating metrics like the *number of visits, time on site*, the number of *page views per visit*, etc. "*Referral*" data can be used to analyze traffic levels from expected and unexpected sources, or to gauge channel effectiveness in advertisement tracking.



- Figure 2: How HTTP request header displayed in Google Chrome browser

- **User action data**: it is mostly keyboard and mouse actions made by users, e.g. user's input of search parameters, and mouse actions like cursor coordinates, movements, clicks. It also includes application specific action such as playback of video or audio, bookmarking, etc. User action data also includes device specification information, e.g. display resolution, CPU model, or any other information about user's settings not restricted by security policies.
- **Network level and server generated data associated with HTTP requests:** this data is not a part of the HTTP request, but it is required for successful request transmissions. Most of this information is exchanging on the TCP/IP level, thus hidden from the user and logged by the web server. Server generated details used for internal reference and recorded in the *server log* files. The log file, typically, records transferred files size, transfer time, server IP, request ID, etc.
- **External data:** any other data gathered during user activities on the web site, like registration, search history, income from advertisement traffic, etc. This is any data, which can be associated with a specific web page. This data could also come from the indirect indicators, e.g. geo-location, revenue generated by the web site, various data gathered by the third party data providers.

1.1.2 – Basic means of data mining of the typical Web Analytics tools

There are three major methods: web server logging, page tagging, and the most recent method – the application logging. The first and the oldest method for data collection is the web server logging, appeared at the same time when World Wide Web was invented. This method is based on data collection on the server's side, where the web site is hosted, it records HTTP headers and some of the server's activities into a textual log file, typically they are: *server IP*, *date* and *time*, *HTTP request command*, *response stat*us, and *response size*, e.g. a common log file implemented using Apache Web Server version 2.2 is shown in figure 3.

Additional information, such as *HTTP headers, process id, scripts, request rewrite*, etc. can be logged in numerous proprietary formats, called the "*Extended log file format*". After emerging of the first web crawlers and search "bots" along with web proxies and dynamically

assigned IP addresses for large companies and ISPs, it became more difficult to identify unique human visitors on the website.

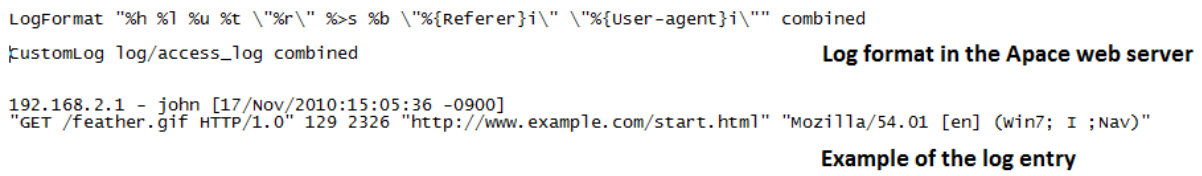


Figure 3: Example of the log file format in the Apache web server

Log analyzers responded by tracking visits with *cookies*, and by ignoring requests from known web crawlers. The use of web cache also makes a problem for log file analysis, because if users revisits a page, the second request will be retrieved from the browser's cache, and logging web server will receive no request. This means that the user's path through the site is lost. Caching can be passed by configuring the web server, but this can result in reduced performance for the users and increased calculation load on the server side, which is undesirable or even inacceptable in some cases. Today log file analysis has its own advantages, such as:

- **No changes to the website required:** each web server normally already produces log files, so the raw data is already available.
- **Privacy and security:** all the data are stored on the company's own servers and is a standard, rather than a proprietary format, which makes it easy for a business to switch web analytics software if needed, facilitating the use of the different software.
- **Log files:** contain information on visits from the search engines, which generally do not execute any logging scripts on the web page, and therefore is not recorded by the page tagging. Although these actions should not be reported as part of the human user’s activity, but it might be a useful information for the search engine or traffic analysis.
- **No additional DNS lookups or TCP slow starts:** there is no external server calls, which can slow down web page load speed, or result in uncounted page views.
- **Transactions:** the web server records every action it makes, e.g. playing video files and content generated by scripts hosted on the web page, and does not rely on the visitor’s browser actions.

Finally, economic factors of the log file analysis include:

- **Local performance:** *log file* analysis usually performed locally, which means the owner somehow need to manage this growing data by his/her own means, which makes additional load on the IT department.
- **Software purchase:** company needs to purchase log file analysis software, which imposes some limitations because vendors may introduce maximum annual page view limit with additional costs to process additional information. In addition to commercial offerings, several open–source log file analysis tools are available free of charge.
- **Data storage:** to perform *log file* analysis, a company has to store and archive its own data, which increases in volumes dramatically and very quickly, resulting in purchasing more storage space. In recent years, cost of the hardware dropping each year, but costs connected with the maintenance of the IT personnel, responsible for hardware servicing, is rising.
- **Maintenance:** for the log file analysis, the company needs to maintain logging software, including updates, security patches and of course cost for the IT department supporting the system.

Web page tagging is the second and more recent method, which uses client-side programs such as *embedded scripts*, browser *add–ons* and *plug–ins*. Page tagging de facto has become a standard in the modern web analytics. For example, using tracking method with JavaScript, a piece of JavaScript code is embedded in the web page and tracks different user's activity, storing information in the cookie files. Then information is sent to a processing server, which might be not the same server that hosts tracked web site, e.g. the most widespread web analytics software - Google Analytics and open-source Open Web Analytics actively uses page tagging technology. For many companies and individuals, it has become a major type of usage of their web data collections. General advantages of the page tagging are:

- **Instant start:** page tagging statistics is activated by opening the web page in user's browser, indicating that the web client runs the tagging scripts. If the web page is cached, the server will not count it, because cached pages can constitute a considerable part of all page views.
- **Data:** is gathered via the "tag" component on the page, usually written in JavaScript, though Java, ASP.NET or Flesh can be used instead. Ajax (a set of Web development

techniques using many Web technologies on the client side to create asynchronous Web applications) can also be used in conjunction with a server–side scripting language, such as PHP, to manipulate and store data in the database, enabling complete control over how the data is represented.

- **Script:** may have access to the additional information about Web client or user, which is not sent in the query, e.g. user's screen resolution or prices for goods he/she purchased.
- **Page tagging:** can report on events, which do not involve a request to the Web server, such as interactions within Flash movies, partial form completion, mouse events, such as *onClick, onMouseOver, onFocus, onBlur* etc.
- **Cookies:** *page tagging* service manages the process of assigning cookies to the visitors, with log file analysis, but the server has to be configured properly in order to do this.
- **Availability:** *page tagging* is available to companies and users who do not have their own web servers.

The third method of data collection is called *application level logging*. In recent times, it has gained wide popularity due to extreme usefulness for the marketing research. Application level logging is closely connected with an application, making it a functional feature of the application itself. This extends possibilities of the traditional web analytics, which now focuses not only on generic HTTP requests and user's actions. The web application can be anything from a small retail shop to a big social networking service. Each of these applications now has its own unique data, e.g. user's registration information, product prices, metadata, etc. that data is collected beyond generic web requests but by user's actions. For example, Microsoft SharePoint 2010 provides framework specific analytics data, like usage of templates and web parts.

1.1.3 – Key metrics used in typical Web Analytics tools

It exists many metrics, which are used for the web analysis. Each software package could have its own metrics, next will be considered only the most commonly used ones [Kaushik, 2009] and those, which will be used in the WARES recommender system:

- **Hit:** a request for a file from the Web server. The number of hits received by the website is frequently cited to assert its popularity, but the total number of visits or page views provides a more realistic and accurate assessment of popularity.
- **Page view:** a request for a file, or sometimes an event such as a mouse click, that is defined as a page in the setup of the web analytics tool.
- **Event:** a discrete action or class of actions that occurs on the website. A page view is a type of event. Events also encapsulate clicks, form submissions, key press events, and other client–side user actions.
- **Visit/Session:** a visit or a session defined as a series of page requests or in the case of tags, image requests from the same uniquely identified client. A *unique client* is commonly identified by the IP address or a unique ID that is placed in the browser's cookie. Data collectors and analysis tools have no reliable way of knowing if a visitor has looked at other sites between page views; a visit considered "a single" visit as long as the events such as page views, clicks, whatever being recorded, are within 30 minutes or less interval. Note that a visit can consist of one page view, or thousands. A unique visit's session can be extended if the time between page loads indicates that a visitor has been viewing web pages continuously.
- **First Visit/First Session:** or "*Absolute Unique Visitor*", in some web analytics tools. A visit from a uniquely identified client that has theoretically not made any previous visits. Note that the *first visit* label is not reliable if the site's cookies have been deleted since their previous visit.
- **Visitor/Unique Visitor/Unique User:** the uniquely identified client that is generating page views or hits within a defined time period, e.g. day, week or month. A uniquely identified client is usually a combination of a device and a browser. The identification is usually via a persistent cookie that has been placed on the device by the site page code.
- **Repeat Visitor:** a visitor that has made at least one previous visit. The period between the last and current visit called visitor "*recency*" and is measured in days.
- **Return Visitor:** a *unique visitor* with activity consisting of a visit to a site during a reporting period and where the "unique visitor" visited the site prior to the reporting period. The individual is counted only once during the reporting period.

- **New Visitor:** a visitor that has not made any previous visits.
- **Impression:** the most common definition of "Impression" is an instance of an advertisement appearing on the viewed page, but most measures of impressions do not necessarily mean an advertisement has been viewed.
- **Single Page Visit/Singleton:** a visit in which only a single page is viewed or a "*bounce*".
- **Bounce Rate:** the percentage of visits that are single page visits.
- **Exit Rate / % of Exit:** a statistic applied to an individual page, not a website. The percentage of visits seeing a page where that page is the final page viewed in the visit.
- **Page Time Viewed/Page Visibility Time/Page View Duration:** the time a single page or a blog, Ad Banner, was viewed. On the screen is measured as the calculated difference between the time of the request for that page and the time of the next recorded request, note that if there is no next recorded request, then the viewing time of that instance of that page is not included in reports.
- **Session Duration/Visit Duration**: an average amount of time that visitors spend on the site each time they visit. This metric can be complicated by the fact that analytics programs cannot measure the length of the final page view.
- **Average Page View Duration:** an average amount of time that visitors spend on an average page of the site.
- **Active Time/Engagement Time:** an average amount of time that visitors spend actually interacting with content on the web page, based on *mouse moves, clicks, hovers and scrolls*. Unlike *Session Duration* and *Page View Duration/Time* on Page, this metric can accurately measure the length of engagement in the final page view, but it is not available in many analytics tools or data collection methods.
- **Average Page Depth/Page Views per Average Session:** page depth is the approximate "*size*" of an average visit, calculated by dividing the total number of page views by the total number of visits.
- **Frequency/Session per Unique:** it measures how often visitors come to a website in a given time period. It is calculated by dividing the total number of sessions or visits by the total number of unique visitors during a specified time period, such as a month or year. Sometimes it is used interchangeably with the term "loyalty".

- **Click path:** the chronological sequence of page views within a visit or a session.
- **Click:** a single instance of user's action following a hyperlink from one Web page to another.
- **Site Overlay:** report technique in which statistics of clicks or *hot spots* are superimposed, by physical location, on a visual snapshot of the web page.

The common type of analysis, which could be performed by most modern web analytics tools, are the *dimensional analysis*, which involves metrics described above, and other derived metrics, aggregated at the different levels. For example, we can use dimensional analysis to answer the question: "how many visits per month, per day, per period?". Dimensional analysis is the fundamental part of other analysis types and reports. The other most common types of analysis are:

- **Trends analysis:** overlooks the data along the time dimension and shows the chronological changes of the selected metrics. For example, the data can show how the percentage of the mobile client access has changed for the past two years.
- **User interest/attention analysis:** is an "internal" page analysis, which analyzes user's attention to certain web page details during the browsing process. It uses embedded script to track user's mouse movements and actions, and shows results in a form of a color matrix or "heat map", where "cold" areas colored in blue or purple indicating a low level of attention and "hot" color areas such as orange and red indicating a high level of user's attention. It can also show how far down visitors scroll the page. Analysis of "popularity" areas of attention helps to develop content placement strategies. For example, it could help to determine where the advertisement should be placed to have the most attention from the users.
- **Cohort analysis:** is a subset of behavioral analytics that takes the data from a given dataset, e.g. an e–commerce platform, web application, or online game, and rather than looking at all users as one unit, it breaks them into related groups for analysis. These related groups called *cohorts*, usually share common characteristics or experiences within a defined time–span. Cohort analysis allows a company to see patterns "clearly" across the life cycle of a customer (user), rather than slicing across all customers blindly without accounting for the natural cycle that a customer undergoes. An example of the

cohort analysis is an owner of the online store, who may only be interested in customers who signed up in the last two weeks and who made a purchase, in this case, this group of users is an example of a specific cohort.

- **Distribution analysis:** explains metric values by building various charts and count tables. Values are usually calculated as percentages of the total by one or more dimensions. It is often used to analyze the number of visitors per period and distribution of preferences in client profiles. For example, the percentages of visits from the different time zones per month may give information about clients shopping habits. Other commonly used dimensions for this type of analysis are: geo location, operation system version, device type, referral source, etc.
- **Clickstream analysis:** also called a *click paths*, it analyzes the navigation path of the user through a website. A clickstream is a list of actions for all web pages viewed by a user, represented in the viewing order. Applying clickstream analysis may help to improve website's design and overall usability.
- **Funnel Reports:** using a series of events that lead towards a defined goal, e.g. from user's engagement in a mobile application to a sale in an e–commerce platform or advertisement to purchase in online advertising. The funnel analyses is an effective way to calculate conversion rates on specific user behaviors. This can be done in the form of a sale, registration, or other intended action from an audience. The origin of the term funnel analysis comes from the nature of a funnel where individuals will enter the funnel, yet only a small number of them will perform the intended goals.
- **Conversion analysis:** is one of the key analyses in e–commerce and other sectors. The conversion rate is calculated by dividing the number of completed targeted actions, e.g. purchases, by the number of unique users visited the site. All web analytics providers strive to improve conversion tracking. For example, Google Analytics provides reports that show what campaigns, sources, or channels have contributed to a visitor's multi–visit conversion.
- **Performance analysis:** helps to reveal website performance issues, such as loading time or linking errors. For example, after a website redesign, indirect traffic volume needs to be watched. If there is less indirect traffic, then some links from other sites and/or bookmarks were potentially broken after the redesign.

- **Engagement analysis:** is one of the most frequently used analyses in the industry. It measures the following factors: “how many pages were visited per session?”, “what is the duration of a visit?”, “how often new visitors become returning visitors?”, “how often visitors return to the site, term *loyalty*?”. The goal of the visitor engagement analysis is to find out why the multitude of operations performed on a website did not end in conversion.

1.2 – Basic concepts about recommender systems

The typical definition of the *Recommender System,* as was described by Peter Falk [Falk, 2015] - a system that applies data mining techniques and some prediction algorithms in order to predict user’s interest on certain items, information, products or services among the tremendous amount of the similar items available on the market. In modern reality, the vast growth of information on the Internet adds new challenges for recommender systems, which are: producing accurate recommendation, coping with huge amounts of data, handling many recommendations efficiently with the growing number of participants in the system. Each recommender system has its own unique structure, but most features are shared between every recommender system. It was proposed using the following dimensions to describe a recommender system: *Domain, Purpose, Context, Personalization Level, Whose Opinions, Privacy and Trustworthiness, Interfaces, Recommendations Algorithms* [Falk, 2015].

- **Domain:** is the type of content, which will be recommended. For example in the IMDB, it is movies, series and actors, but it can be anything: cars, e–learning courses, job listings, food, books, hotels, etc. in the case of WARES it is web analytics tools. Domain is an important because it provides hints on what would you do with recommendations.
- **Purpose:** defines why the recommender system was created and the goals which it pursues. For example, a typical end–user’s purpose of the eBay recommendations is to find a product that user wants to buy. The purpose for the recommendation provider (eBay) is ultimately to make customers pay for their purchases by having a certain % of the final purchase price. Another example of a purpose is to give an information or to help, to entertain, to educate the user, but in most cases, the purpose is - to sell more. In

the case with WARES the purpose is to give a single or few recommendations for the user that typically arrives once or a few times and expects a "good" recommendation.

- **Context**: is the environment in which the consumer receives a recommendation. For example, Netflix delivers service on many different platforms. The device, which customer is using is the context. The context is also could be the current location of the user, what time it is, and what the user is doing. Does the user have time to study the suggestions or a quick decision is needed? Context could also be the weather around the user. Consider a search for a restaurant using Google Maps. Is the user sitting at home and looking for a good restaurant or he is standing on the street and it just started raining? In the first scenario, the best response would be about good quality in a bigger radius, while in the second scenario, recommendations would ideally contain only the nearest place where you can drink and eat while the rain passes. In the case of WARES the context of the recommendation expected to be an office or home place, so the user will have some time before taking the final decision.
- **Personalization:** recommendations can come at many levels of personalization, from using basic statistics to looking at individual user's data, three levels of personalization could be highlighted:

  a) Non–personalized: a list of the most popular items is considered non–personalized recommendations: it is expected the current user might like the same items as most others do. Non–personalized recommendations also include showing things on sale or ordering items by date, such as showing the lowest price items first. Everybody who interacts with the recommender system receives the same list of recommendations, e.g. if user visiting Amazon.com as not registered (anonymous) user Amazon shows (recommend) items that are currently viewed by other users.

  b) Segment–personalized: is when you divide users into groups. There is many different ways to segment users. It can be done by age, by nationality, by specific patterns, such as entrepreneurs or students, car drivers or bike cycle riders. Example of segment–personalization can be a concert ticket selling system that would recommend concerts based on the country or city. If user listening music using a smartphone, the system might try to deduce whether a person is out for exercising, by using the GPS and seeing that, the device is moving. While if it is stationary and considered as "at home", then the

consumer is probably sitting on the sofa and the relevant music might be different, the recommender system does not know anything personal about you as a person, only as a member of a group. Other people who fit into the same group will get the same recommendations.

c) Personalized: a recommendation, which is based on the data about current user. Based on how the user interacted with a system in the past, as well as data on other forms of user's behavior. This will generate recommendations specifically for this user. Most recommender systems using segments and popularity when creating personalized recommendations. For example, if a user looking for car parts at the bottom part of the web page of each eBay's auction you can see a section "More Parts for your vehicle", it is a personalized recommendation. Netflix is an extreme example of personalized, it will apply different types of recommendations, but so far, there are only a few examples like Netflix where everything is personalized. On YouTube you will see the list of videos, which is recommended based on your list of previously viewed videos. In the WARES given recommendations will be *non–personalized*.

- **Whose options:** in other words the area of experts, popular feature used on many websites in the past, for example "Canadian Tire recommends this brand of tires for your vehicle", but is rarely used nowadays. Majority of the recommender systems integrated into websites now uses the opinion of the *"masses"*. In the past were expert systems, where pre-recorded variants from some experts was combined to recommend products such as wines or books, where it was, "accepted" that someone should be an expert to understand what is "good", but modern recommender system, has no specific experts who decide, an algorithm representing opinion of the "masses" does all the work.
- **Privacy and trustworthiness:** nowadays it is a serious concern! How well does the system keep user's personal data, how the collected information is used, how it is stored, etc.? The privacy is most likely the question of security measures, taken by recommender system owners, for storing personal user data complying with local laws about user's data. Another aspect is trust, for example, it is quite common that you will have to pay money for your pension plan, which is handled by commercial banks. Often these banks will have different kinds of retirement savings schemes. A system, which should recommend these options, should have very strict rules for privacy. Imagine a user,

filling some data to get a pension plan recommendations, and describing that he has back problems, and a minute later receiving a phone call from a chiropractor with great offers to handle your exact problem. Or even worse, you buy a special bed for people with backbone problems, and an hour later, you receive an email that your health insurance premium has gone up. Many people think that recommendations are manipulation, because they present choices that a customer is more likely to pick than if they were offered as a random selection. Most shops are trying to sell more, so the fact that some shops are using recommendations to sell more makes people think that they are being manipulated. In fact, any form of filtering is a manipulation, but if that means watching a film that would entertain rather than bore, then it is okay. The trustworthiness is about how much the consumer will trust recommendations instead of considering them as commercials or attempts to manipulate them, it means if a user takes recommendations seriously, the system could be considered trustworthy. In WARES trustworthiness cannot be measured for several reasons: it is new, nobody except me tested it, the system has no intention to store any personal user's data, like names, addresses or credit card numbers, only certain survey data about user's needs in web analytics, which could be deleted after session ends.

- **Interface** of the recommender system is considering what kind of input and output it will produce. For example, Netflix enables the users to enter taste preferences by rating content, and add preferences on genres and topics. Netflix outputs recommendations in many ways. Netflix estimates predictions, they provide personalized suggestions, they show popular items, which normally is shown in the form of a "*top 10*", but Netflix even personalizes that. It also provides suggestion based on the content you watch. Types of output could be predictions, recommendations or filtering. If the recommendations are integrated as a natural part of the page, it is called *organic presentation*. The rows shown on the Netflix website are a good example of the organic recommendations, they do not seem like a recommendations, it looks like an integral part of the web site; while the recommendations shown on Amazon are considered as non–organic personalized recommendations, and are shown in the field "Customers who bought this…". Example of organic and non–organic recommendations is shown on the figure 4. The WARES is

designed to be a web application, it means the user could use WARES via his/her favorite web browser and will receive *non–organic* personalized recommendations.

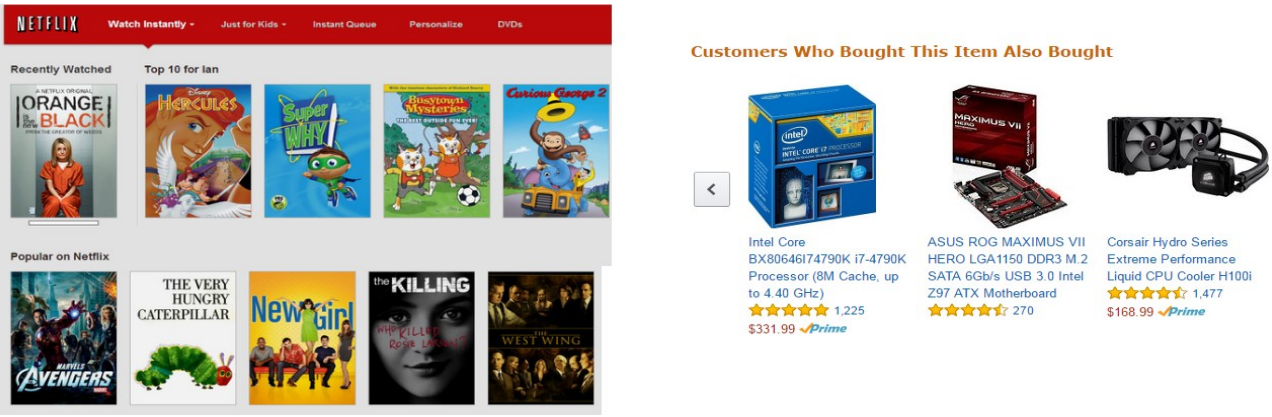


Figure 4: Organic (right) vs. non–organic recommendations representation

- **Algorithms:** there are essentially two groups of them, based on what data the one uses to make recommendations. Algorithms that uses the user's data are called collaborative filtering. Algorithms that uses content metadata and user profiles to calculate recommendations are called content-based filtering, more detailed they will be considered in the next section and in the chapter 2.

Some recommender systems will try to explain given recommendations, which is called the *white box* recommenders. For example, Amazon shows following text "Customers who bought this item also bought…", which could serve as an explanation why this item was recommended to you, while recommender systems which do not try to explain are called the *black box*, e.g. Netflix just showing covers of recommended movies, without any explanations. Examples of such cases are shown on the figure 4. This is important to consider, when choosing the algorithm, since not all of them provide a clear path back to a reasoning in the prediction process. Deciding whether you want to produce a white box recommender or black box is quite important, since it can put quite a lot of restraint on which algorithms you can use and it also affects the level of the trustworthiness for your recommender system. The less your system needs to explain the more simple the algorithm will to be. The WARES will implement some of the *white–box* features.

1.2.1 – Basic approaches to solving recommendations problem today.

There are two major groups of algorithms, differing from each other by the data they are using to produce recommendations. Group of algorithms that uses collected data of user's behavior are called collaborative filtering. Algorithms that use data from the description of the item and from the user's profile preferences are called content–based filtering.

Collaborative filtering: a set of methods which are based on collecting and analyzing a large amount of information on users' behaviors, activities or preferences and predicting what users will like based on their similarity to other users. Figure 5 illustrates a simple way of collaborative filtering process. The outer set is the full catalogue. The middle set is a group of users, which consumes similar items. A recommender system will now recommend items from the *"segment of user's preferences"* set, assuming that if users liked the same items as the current user, then a current user will also like other items in this group. This means that a current user is *matched* with other users. Then a gap of content, which the current user is missing, will be recommended, i.e. the part of the "*middle sized"* set which is not covered by the set representing what the current user likes. Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past.

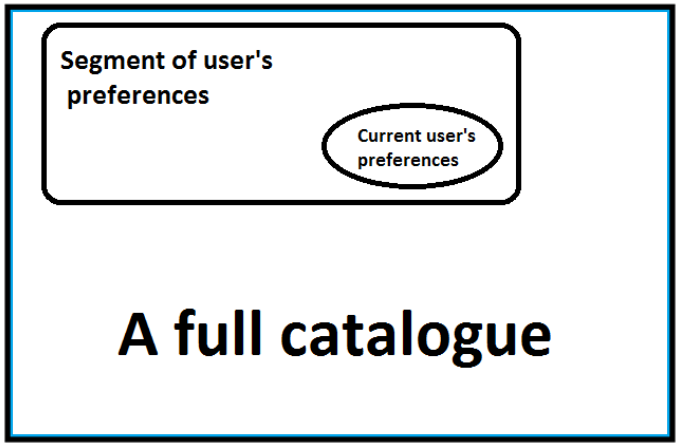


Figure 5: Collaborative filtering process

Many algorithms are used for measuring user similarity or item similarity in recommender systems. For example, the *k–nearest neighbors* (k–NN) approach and the *Pearson Correlation*, see chapter 2 for details.

Content based filtering: is all about looking at the relationships between the items and original user's profile preferences, based on objects and items user interacted in the past a "second" user's profile is built to indicate the type of items this user likes. The system then constructs a profile for the each user, which contains categories of the content. To create this "second" user profile, the system mostly focuses on the original profile of the user's preferences and a history of user's interaction with the recommender system. Therefore, this method uses an item profile i.e. a set of discrete attributes and features, characterizing the item within the system. The system creates a content–based profile of users based on a weighted vector of item features. The weights denote the importance of each feature to the user and can be computed from individually rated content vectors, using a variety of techniques. Simple approaches use the average values of the rated item vector while other sophisticated methods use machine learning techniques such as *Bayesian Classifiers*, *cluster analysis*, *decision trees*, and *artificial neural networks* in order to estimate the probability that the user is going to like the item, for see chapter 2 for details.

1.3 – Comparative review of some existing Web Analytics tools

Today a wide range of tools for the web analysis was developed, to better understand their variety, and what user should expect from them, a comparative review of some notable web analytics tools was made. All these web analytic tools were present on the market when this review was done, year 2016. A table at the end of this section shows them all together, comparing some of their key features.

1.3.1 – Google Analytics

Beginning with the major favorite [w3techs] on the market of the Internet–related services, a Google Corporation with its well–known "Google Analytics" [Google Analytics,

2016], next GA. It is a widely popular tool because of its integration with Google search engine, wide variety of metrics and relatively simple interface and fast setup, consisting of three simple steps, which are shown on the figure 6.

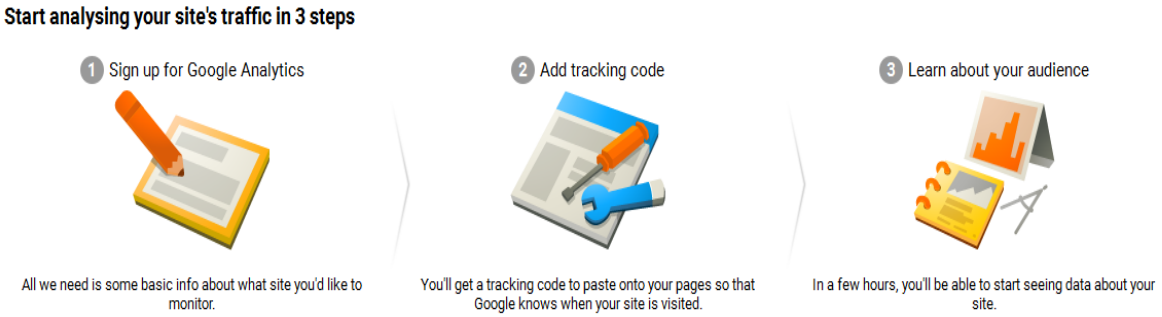


Figure 6: Google Analytics setup, (screenshot from Google.com)

Google launched GA service in November 2005 after acquiring Urchin Software Corporation, which was widely known by its web analytics product "Urchin" applying log file analysis on the server side, designed to show traffic information on the website based upon the log data. GA offers a full range of analysis for Web, but as always with Google in exchange of your privacy, it means that Google will have access to some of the user's personal data such as geo-location, your search history, click history, etc. After completing the first step, consisting of entering the URL for the website, for which statistics will be gathered. Desirable name for analysis report and some other data, e.g. reporting time zone, and some extra features, user will get a Tracking ID and universal analytics tracking code, which user could insert into every web page that he/she wants to track, example of the JavaScript code and tracking ID for GA is shown on the figure 7.

Tracking ID

UA-87827507-1

Status

No data received in past 48 hours. Learn more

Website tracking

This is the Universal Analytics tracking code for this property.
To get all the benefits of Universal Analytics for this property, copy and paste this code into every web page that you want to track.

```
<script>
  (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
  (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
  m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
  })(window,document,'script','https://www.google-analytics.com/analytics.js','ga');

  ga('create', 'UA-87827507-1', 'auto');
  ga('send', 'pageview');

</script>
```

Figure 7: Google Analytics tracking ID and tracking script, (screenshot from Groogle.com)

For example, this tracking code was inserted into one of the web pages hosted on the University of Montreal web server for students. A resulting report for one–month period from mid–October 2016 to mid–November 2016 is shown on the figure 8. This is a standard report about visitors, representing total number of *visits, average session duration*, *country of user's origin*, etc.

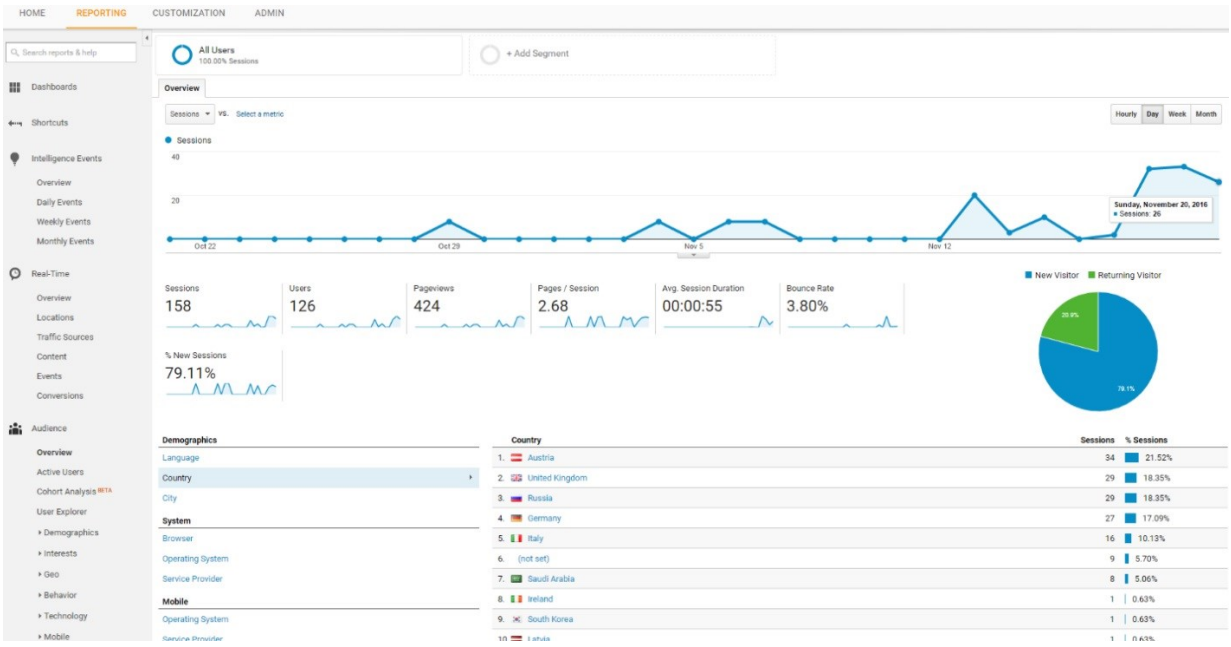


Figure 8: Google analytics results for 1 month, (screenshot from Google.com)

By customizing this report, many other *metrics* and *goals*, could be added, see section 1.1 for details about used *metrics*. Typically, a *goal* is an action that the site owner wants a user to complete, e.g. viewing a particular number of pages, going to a specific page, clicking on a specific button, clicking a link, making a purchase, and so on. Goal completion is meeting the conditions set in the goal parameters. A session in which the goal was completed is called a *conversion*. A single conversion may include multiple completions of the same goal. The ratio of conversions to the total number of sessions is called the *conversion rate*.

Some major features of Google Analytics are:

- **Cloud hosting** with computational power from the leading data centers on Earth.
- **Documentation**, lots of documentation about it, so anyone could learn how to use it.
- **Video performance,** allows to measure your video performance using YouTube service.
- **Integration** with AdWords, DoubleClick Bid Manager, AdSense, AdMob, and other advertising service for businesses wanting to display ads on Google.
- **Support for all devices**, running Windows, Linux, Android, iPhone/iPad and Mac.
- **Cutting edge technology** from the leading IT company on the market for Internet services.

Some disadvantages of Google Analytics are:

- **Privacy** – it is a major tradeoff for receiving free quality service in web analytics from Google, as by using GA, you automatically agree to share your data with Google.
- **Support is limited** to a help center and user forum unless you hire support from a certified partner.

Standard and customized reports could be distributed by any accessible time periods, e.g. day, week, month, and what is very important in the age of social networks GA could detect *user bounces* from social networks such as Facebook or Twitter and evaluate user's social network preferences, e.g. user's most popular categories in a social network. GA also providing *Cohort analysis*, see section 1.1.3 for details, which helps to understand the behavior of component groups of users apart from total population of users. It is very much beneficial to marketers and analysts for successful implementation of marketing strategy. GA is a "*freemium*" service, which means it is provided free of charge, but money (premium) is charged for proprietary features, functionality, or virtual goods, for example a user can have 100 GA site

profiles. Each profile generally corresponds to one website. It is limited to sites, which has traffic of fewer than 5 million page views per month, roughly it is two page–views per second, so to have more statistical data the one must pay for maintenance of additional profiles, unless the site is linked to a Google AdWords campaign. As reported by Web Technologies Survey website [w3techs], which is making surveys for various types of technologies used on the web, market share claimed by Google Analytics is around 54% of all websites in the world.

1.3.2 – Open Web Analytics

An *open source* web analytics software "Open Web Analytics" package [OWA, 2016], which anybody can use to track and analyze how users use websites and applications. OWA is licensed under the GPL license and provides website owners and developers with easy ways to add web analytics to their sites using simple JavaScript, PHP, or REST based applications, it has nice and relatively clear user interface, which is shown on the figure 9. Like all open–source software its source code is *open,* so this mean if the one knows something in the matter of programming he/she can modify this web analytics tool for his/her own needs.
Major benefit features of OWA are:

- **No restrictions** on the amount of user data, it can be used for multiple URL resources.
- **Information** about last visit, individual and detailed reports on the last site visitor with its locations, type of browser, pages viewed, visit duration, referral, etc.
- **Click–stream, heat click and click tracking maps**, which intend to record user's cursor movements, where user click the most, track where exactly on the site user click's.
- **Suppor**t for WordPress and MediaWiki plugins.
- **PHP** applications support.
- **Cost – Free!**
- **Privacy** means all the data is stored on your own server.

Major disadvantages of using OWA are:

- **Time** – it takes some time to install and figure out how to use, as always with most GPL applications.
- **No mobile** applications support.

- **No export** – no support for file export function from the application interface, you can only do this from the database.

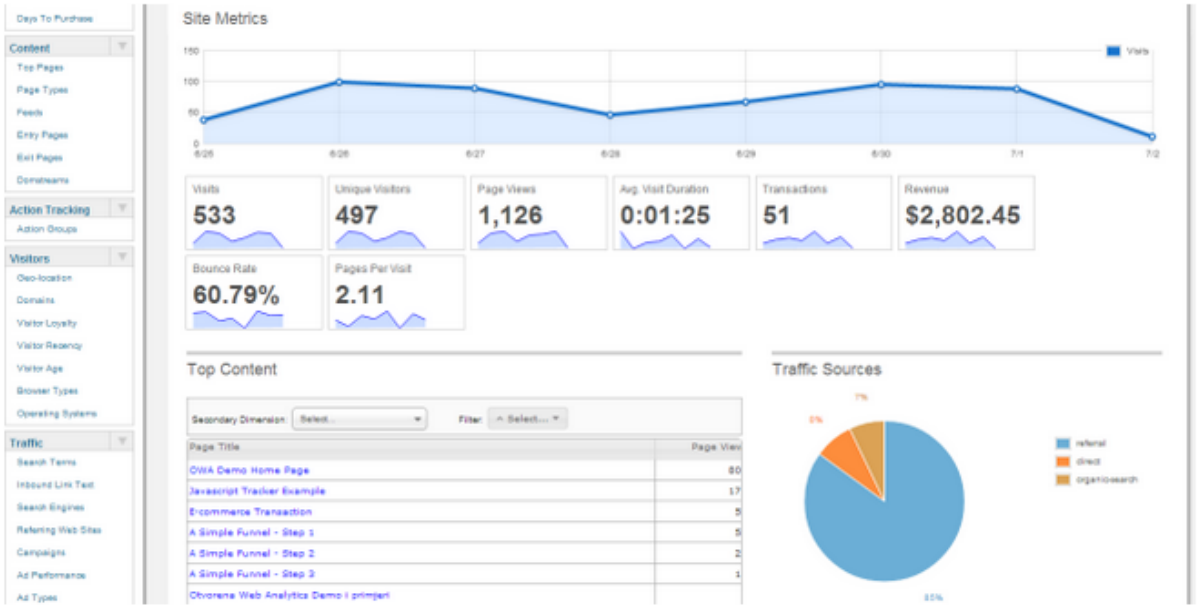


Figure 9: Open Web Analytics dashboard, (screenshot from Open Web Analytics website)

1.3.3 – Yandex Metrica

Another powerful web analytic tool is Russian Yandex Metrica [YM, 2016], in the post–soviet countries of Eastern Europe and in Russia it considered as a major adversary of Google Analytics. It is connected with Russian search engine, which is also called Yandex. By registering a user account for Yandex, called "Yandex passport", a user automatically gets free access to Yandex Metrica services. To configure YM first the one should register with Yandex, it is like a Google account for GA, but Yandex claims that it will not use user's personal information to gain profit, as Google does. Second log in into your account, go to YM page, insert website's URL, the one wants to track and get tracking HTML code for it, and it is done. Now the user can start creating his/her own dashboards and charts, with the most popular metrics and some unique features only available for Yandex Mertica users, such as Yandex Direct, which is actually an analogue of Google AdWords. Hence, this web analytics tool was

developed in Russia, it also provides an English user's interface, so potentially it could be used by US and Canada users.

Yandex Metrica provides four major conversion types:

- **Goals** related to tracking someone landing on a specific page URL.
- **Achievements** – plan for achieving a specific amount of page views.
- **Events**, such as clicks on specific buttons.
- **Multistep goals** where you can combine up to five page URLs or events.

One account in Metrica allows the one to track multiple websites, each having its own set of goals. YM offers an asynchronous code by default in code settings. This type of code does not block or influence the loading speed of your website. It also does not matter where the code is placed, e.g. in the header, body or the footer, even if someone decides to leave the page before it loads completely it will still be classed as a "*visit*". Visitor's actions are reflected in Metrica's reports between 30 seconds and 5 minutes after the actions, whereas all the other statistics are updated every 30 seconds. All the data is displayed in the table, or on the dashboard, as well as in a variety of neatly presented graphs, interface of the YM is shown on the figure 10, alongside with a simple dashboard.

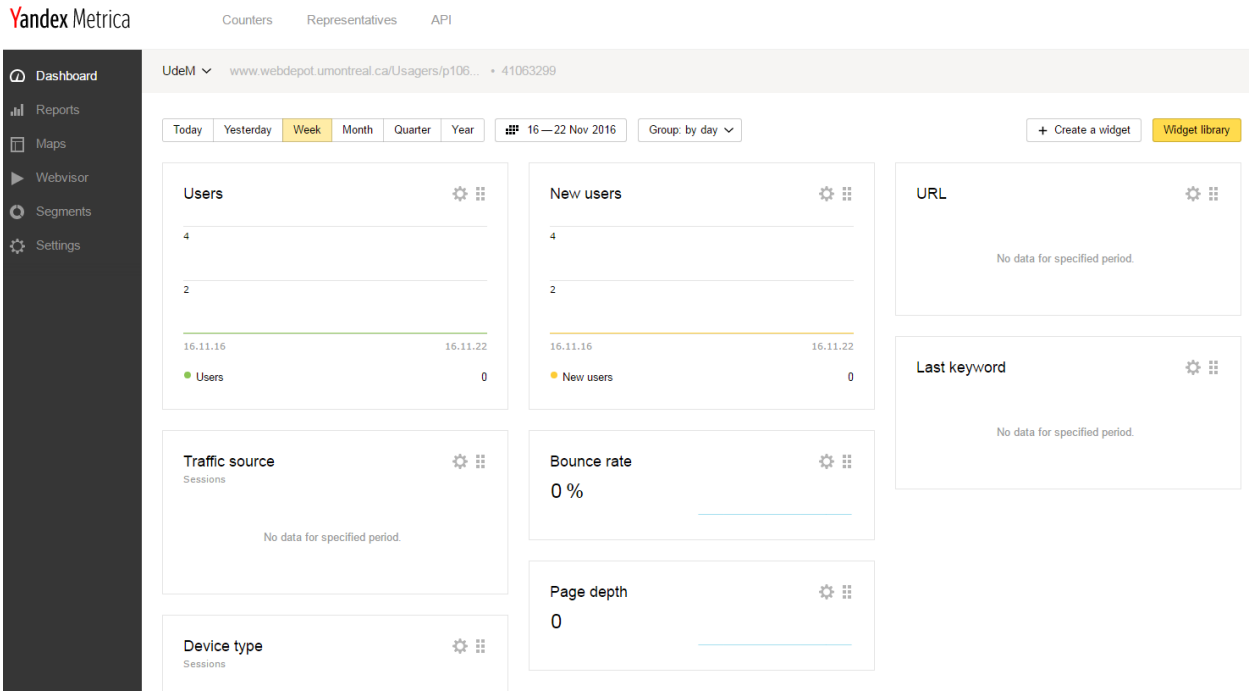


Figure 10: Yandex Metrica dashboard, (screenshot from Yandex website)

Yandex Metrica offers following major reporting features:

- **Traffic and source reports**, including traffic source by region, traffic behavior on your website (number pages viewed, bounce rate, time spent on each page, etc.)
- **Visitors interests**, which are calculated using analytics technology called Crypta. This technology classifies web users based on their previous online actions. Crypta is based on Yandex's own machine learning method MatrixNet.
- **Sources** which leads visitors to your site, e.g. ads, search queries, social networks.
- **Content reports**, which shows statistics on visitor's interaction with the site's content such as entry and exit pages, traffic from external links, file downloads, orders made by visitors and their costs, it is very useful for marketing research.
- **Behavior reports:** if YM user wondered how people behave when they land on his/her website, using Metrica he/she can see a recording of every visit, up to 1000 per day!
- **Analysis of user behavior** during filling of a web form, this tool provides information on the number of views of the page containing the form, the number of interactions with the form, the data on sent forms as well as the video recordings of how people behaved when filling in the form.
- **Click path analysis** with text highlighted in different colors on the map depending on their popularity.
- **Heat maps**, which measure and displays statistics for clicks on the website, also highlighting by colors most clicked areas. Unlike the link map, the heat map shows clicks on all page elements and not just on links, and even further if a user has 3D glasses he/she can see 3D map!
- **Scroll map**, by using the scroll map, it is possible to find out how users of long websites view different elements of the page.
- **Report builder**, allows to create your own report from all the statistics available with a variety of filters and segmentation options to choose from.
- **Offline monitoring**, Metrica can track when your website is down and provides a report on this data. If a user connected his/her Yandex account with Yandex Direct (Yandex

advertising account, analogue Google AdWords), Metrica will switch off your ads to prevent unnecessary traffic spend and it will send you an email saying that it did it.

1.3.4 – Piwik

The Piwik [Piwik, 2016] is an *open–source* web analytics platform developed in France back in 2007. It has paid features called Piwik Pro, which is actually a team of Piwik developers performing customization of the Piwik engine for particular client needs. It is also an interesting fact that Canadian government [Piwik study, 2016] using modified Piwik engine for Fisheries and Oceans Canada as a high–performance and robust analytics tool, other Canadian governmental institutions also employees Piwik for intranet and web sites as an alternative for Google Analytics, because of its "*unsecure*" usage of personal information, which is not acceptable in the governmental sector. Piwik has all major web analytic tools features such as *traffic analysis, click analysis*, etc.

Some major and unique Piwik features are;

- **Self–Hosted**, means it allows a customer deploys analytics engine on his/her own infrastructure and enjoy 100% control and ownership of his/her data and information.
- **Piwik Pro Cloud**, offering full support for customer's web analytics project from Piwik team with secure hosting of your data in the Piwik cloud server, first 30 days are free then you should pay $65 per month.
- **High privacy standards**, Piwik enterprise analytics enables you to comply with EU and local privacy regulations and ensures user's privacy is protected.
- **Unlimited number** of websites and users for statistical data.
- **Multi language** user interface, supporting 53 languages! e.g. on the figure 11 is shown how typical Piwik dashboard looks like in Russian language.

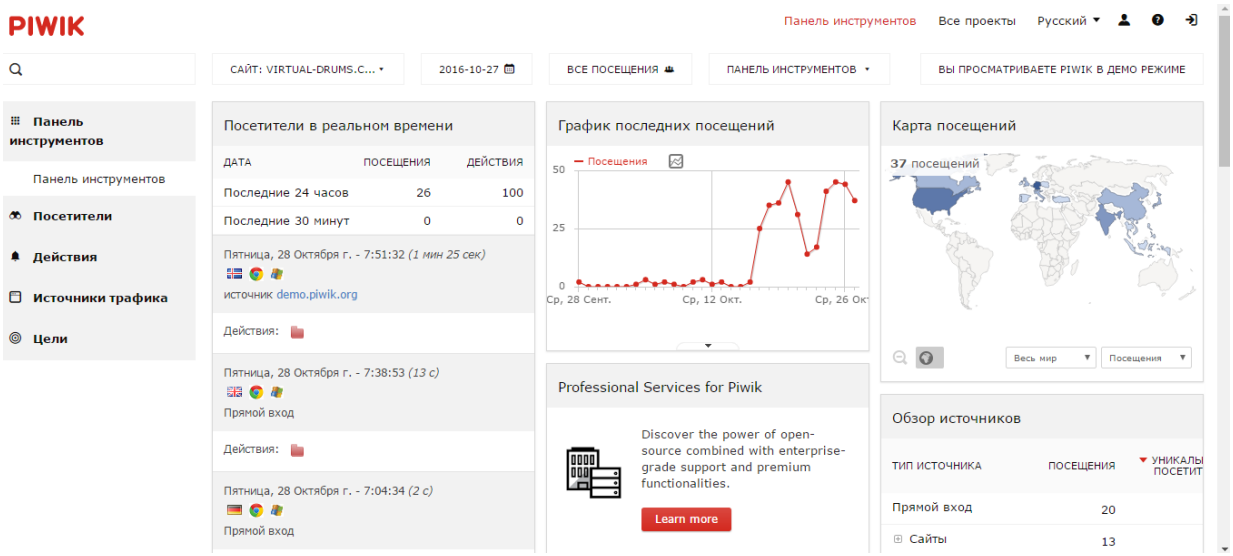


Figure 11: Piwik dashboard demonstrating Russian interface, (screenshot from Piwik website)

- **Tracking of individual users**, allowing site's owner to create a user profile for specific user or group of users to study user's behavior more deeply through the owned website.
- **High customization** offers wide variety of customization options allowing transforming Piwik platform to meet almost any needs, from high security sphere of banking and healthcare to simple website statistics.

1.3.5 – Woopra

The Woopra [Woopra, 2016], is an interesting example of the web analysis tools in the way how it focuses on the marketing. Like any other web analytics software, it can perform all the basic web analytic procedures using script code on the desirable web page. It provides the full array of the statistical information about visitors: *number of page views, traffic, clicks, top referrers, visitor language, browser setting*, etc. But, the key unique feature of the Woopra that it positioning itself as a real–time customer analytics service tailored for use especially in sales and marketing teams. The major difference from all the previous web analytics tools that Woopra uses application interface, not the web interface like majority web analytics tools do, but offering a variety of methods and techniques for tracking user's activities, giving very

detailed information, e.g. which product was added into the customer's basket in real time. Woopra's interface is shown on the figure 12.

Some unique Woopra's features are:

- **Deep user monitoring** and customization, allowing even assigning names for certain users for subsequent identification, using Woopra profiles.
- **Funnel Reports**, allowing using Funnel analytics, see section 1.1.3 for details, to pinpoint where users drop off in the conversion process and to understand why your leads do not convert, which facilitates monitoring on how different segments move through your "funnel" to identify the best and the worst conversion types of the customers.
- **Measures user retention**, allowing increasing customer lifetime by giving suggestions if users continue to do important actions, such as purchases, use your product, or even open your emails and helps the website's owner better understand if users are engaged enough with your offerings to keep coming back.
- **Live chat**, it means as an owner or admin of the website you can even communicate with your website users through the specific interface.
- **Personal tech support** from Woopra's customer service.

Woopra is a proprietary software, but it offers a free account with 30.000 user actions per month and limitation for data storage for 2 months, if you need more you should pay starting from $80 per month or higher depending on your volumes of information.

Disadvantages of using Woopra:

- **Complicated interface:** Woopra application itself has relatively complicated interface.
- **No support** for mobile applications.
- **English** interface only.
- **Storage limit:** even with paid subscription, your data will be stored maximum up to 24 months.

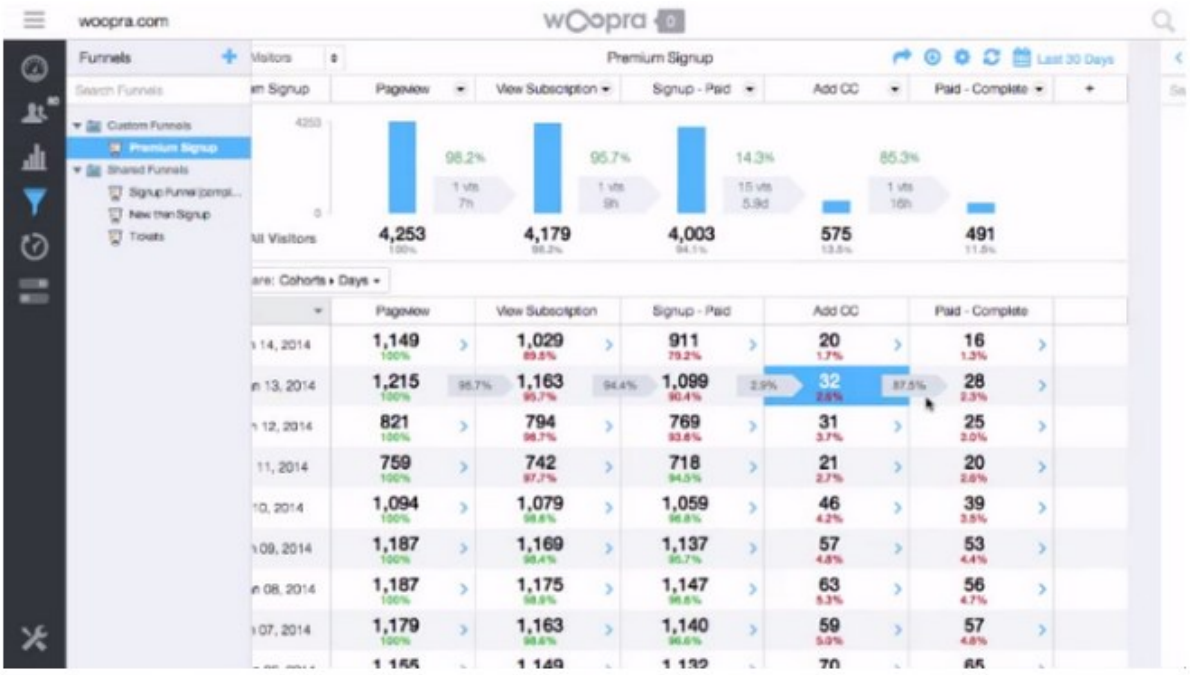


Figure 12: Interface of the Woopra main application window, (screenshot from Woopra website)

1.3.6 – Summary on comparison of web analytics tools

The list of the web analytics products could be endless, but I think I gave an idea what web analytics tools look like and what they are offering for their users. Table 1 highlights the final comparison of all reviewed web analytics tools. It is difficult to compare all the existing properties, the table would be simply too long, with too many features, because of the nature of the whole paper the comparison is not an objective itself, that is why this comparison was made based on availability of the certain properties, which could represent the interest for the certain potential users:

- **Mobile support:** possibility to gather statistical data from the mobile devices like smartphones or tablets.
- **Real–time statistics:** possibility to have statistical data, changing in the real-time.
- **User click–streams:** availability of the click–stream analysis.
- **Conversion methods:** possibility to have multiple sessions in which goals, which will be analyzed, were achieved by the visitors.

- **Free–trial:** show the possibility of using this web analytics tools in a trial mode.
- **$ per month:** this field representing a price, which potential user will need to pay monthly in order to use one of these web analytics tools.

Table 1: Comparative summary of the considered Web Analytics tools

	Open Web Analytics	**Google Analytics**	**Yandex Metrica**	**Piwik**	**Woopra**
Mobile support	no	Android, iOS, Unity	Android, iOS	Android, iOS	no
Real–time statistics	no	yes	yes	Every 5 sec.	yes
User click–streams	yes	no	yes	no	no
Conversion methods	yes	yes	yes	yes	yes
Widespread*	<0.1%	54%	5.4%	1.3%	0.2%
Free–trial	unlimited	unlimited	unlimited	30 days	>30k user actions/month – free
$ per month	Free	"Freemium"	Free	Free/$65/custom	Free/$80

This short comparison concludes this section, in the following we will consider some of the most widely-used methods and algorithms for solving the recommendation problem.

* according to the statistics from https://w3techs.com, December 2016

1.4 – Conclusion

In this chapter, was given an idea in which direction this report will continue. While designing a recommender system, it is essential to know the basics highlighted above. First part was dedicated to the field of the web analytics, describing it as a niche of the software market. To continue it is very important to know these concepts, methods and metrics by which the web analytics software operates, because it is impossible to design a recommender system without knowing the field in which recommendations will be made. Second, it was explained what is the recommender system, how it should work, what are the typical objectives for the recommender system, current trends in the development of the recommender systems and two major approaches for implementing them: content–based and collaborative filtering techniques. Finally, was made a comparative review of some chosen web analytic tools available today (the year 2017) on the market, showing their advantages and disadvantages, which gave us an excellent opportunity to see how different web analytic tools works and how they look in practice. This was an essential starting point for this paper, which gave us a direction where to move next. In the next chapter will be described in details some chosen algorithms and techniques used for creating recommender systems. These techniques will be described from the collaborative and content–based viewpoints, at the same time, focusing on the consideration of those methods that will be used in the WARES recommender system.

Chapter 2 – Methodology

In this chapter will be discussed methods and techniques, which was used for the WARES and some others that are commonly used in the field of the recommender systems development. Most of them was already implemented in such well–known e–commerce projects like Amazon, Netflix and eBay, considering them step by step will allow us to decide which ones are suitable for the WARES recommender system and which are not. Those techniques and algorithms, considering essential to the WARES recommender system, will be explained in details.

The appearance and growth of the online markets have had a considerable impact on the habits of consumers, providing them access to a greater variety of products and information on these goods. While this freedom of purchase has made online commerce into a multi–billion dollar industry, e.g. for consumer goods Amazon.com, eBay.com, for services (watching films, series) – Netflix, but it also made more difficult for consumers to select the products best fitting their needs. One of the main solutions proposed for this information overload problem are recommender systems, which could provide automated and personalized suggestions of products to consumers.

In its most general form, the problem of recommendations is reduced to assigning some rating values for different goods and services, which are not yet known to the potential user/customer, e.g. 5–stars ratings on Amazon.com, is a simple feedback, which users would leave after purchasing the product, or even make a review on 5–star scale. In the case with WARES this *"goods"* are web analytics tools. Obviously, that such an assessment, or rating, could be given based on previous analysis of user's/customer's preferences or any other information on it. After the recommender system predicts ratings for not yet known consumer products those, which receive the highest scores, are recommended to a potential customer. If the recommender system does not possess a certain number of users and ratings, the new ratings, for yet not evaluated products, could be synthetically produced by various methods like *heuristics*, *approximatio*n and various *machine learning* techniques. This "new" ratings are

based on previously received small set of ratings, or if the rating for certain products are unknown (absent) by an *extrapolation* from known to unknown ratings by using:

- **Heuristic rules:** a choice of heuristics that define a utility function, and empirical justification for functional behavior.
- **Utility functions:** various functions, which optimize the parameters of ratings, e.g. a *standard deviation*.

Once the analysis of unknown ratings is done, the user receives recommendations for products or services with the highest ratings from the whole set which was analyzed. Typically, recommender system gives recommendations in two ways: in a form of a set of products most appropriate for the customer or set of customers which are most relevant to the product. Nowadays most recommender systems are built on one of the following approaches:

- **Content–based recommendations** – the customer will receive the recommendation of goods similar to those he/she has chosen previously, e.g. recommendations on Amazon.com representing algorithms used to personalize the online store for each customer, showing programming titles to a software engineer and baby toys to a new mother.
- **Collaborative recommendations** – consumer products will be offered as similar to those, which was chosen in the past by a group of selected people with similar tastes to customer's preferences. For example, recommendations for music on last.fm internet online music service, where you receive recommendations type "users who listening this song also liked this… ".
- **Hybrid methods** – recommendations that combine the two previous methods. In this chapter, combinations of content–based and collaborative filtering techniques will not be highlighted, they are simply too many. Instead will be considered major content–based and collaborative techniques. For example, Amazon is a bright example of a hybrid recommender system, which uses content–based approach for recommending items based on item–item approach and at the same time uses collaborative component to recommend products based on 5–star feedback from other users who already bought similar products.

2.1 – Collaborative filtering approach and possible ways of applications in WARES.

Collaborative filtering is almost all about different rating values and a large number of users who give these ratings. By using collaborative filtering methods various user-specific recommendations of items could be produced based on patterns of user's ratings or customers actions, e.g., clicks and purchase history, without the need for different information about items or users itself. To better understand collaborative filtering approach a good example is needed, here and further let us consider Netflix as such an example. As by its nature, recommender systems based on the collaborative approach rely on various types of input and the most valuable among it is an *explicit feedback*, where users directly report on their interest in products. A movie recommender system Netflix collects star as ratings for movies, while TV series subscribers shows their preferences by hitting thumbs–up/thumbs-down buttons. Frequently *explicit feedback*, is not available, that is why some recommender systems infer ratings from such called *implicit feedback*, which indirectly reflects preferences through observing various user's behavior [Oard and Kim, 1998]. Sources of the *implicit feedback* could be: user's *purchase history*, *browsing history*, *navigation patterns*, etc. For example, a user who watched many films with Clint Eastwood probably likes that actor. Implicit feedback is a valuable source of information for recommender systems where users have not provided enough explicit feedback. Next step, in order to build recommendations, RS based on collaborative filtering need to relate two fundamentally different entities: the *items* and the *users*. There is two most widely-used techniques for collaborative filtering:

- **Neighborhood approach:** focuses on the relationships between items and users, called item–item model, where user's preferences on the items based on ratings of the similar items rated by the same user.
- **Latent factor models:** or matrix factorization, also known as singular vector decomposition, a technique where items and users are transformed into the same latent factor space. The latent space explains ratings by characterizing products and users on factors automatically inferred from the different sources of user's feedback.

Section 2.1.1 describes theory and practical details behind some of the most recent matrix factorization techniques, their relatively high accuracy of prediction also has made them the preferred technique for the Netflix data set.

Section 2.1.2 gives attention to the family of neighborhood methods, their resulting accuracy is close to an accuracy of matrix factorization models, while offering some advantages by lifting the limit on neighborhood size, and addressing temporal dynamics. Explains how to predict ratings using neighborhood techniques and also gives an idea how to measure accuracy, stability and efficiency of user-based and item-based rating prediction models.

Section 2.1.3 considers most widely-used similarity measurement methods and their applications in recommender systems in general and in WARES in particular.

To better understand possible perspectives of application collaborative techniques in the WARES let us consider a typical collaborative filtering model, as was described in the article about collaborative filtering by Yehuda Koren and Robert Bell [Koren and Bell, 2011]. Let us consider that ratings are given for $\boldsymbol{m}$ users and $\boldsymbol{n}$ items, using letters to distinguish users from items: for users $\boldsymbol{u,v}$, and for items $\boldsymbol{i, j, l}$. A rating $\boldsymbol{r_{ui}}$ indicates the preference by user u of item $\boldsymbol{i}$, where higher values mean stronger preference. For example, values can be stars (integers) ranging from 1, indicating no interest to 5, indicating a strong interest. Predicted ratings from known users using the notation $\boldsymbol{\hat{r}_{ui}}$ for the predicted value of $\boldsymbol{r_{ui}}$. The scalar t_{ui} denotes the time of rating r_{ui}. In most cases lots of ratings are unknown, e.g. like in the Netflix database 99% where only a small portion of the movies have been rated by the user. The $\boldsymbol{(u, i)}$ pairs for which $\boldsymbol{r_{ui}}$ is known are stored in the set $\boldsymbol{K = \{(u, i) \mid r_{ui}\}}$, where r_{ui} is known**.** User $\boldsymbol{u}$ is associated with a set of items $\boldsymbol{R(u)}$, which contains all the items rated by $\boldsymbol{u}$. The set $\boldsymbol{R(i)}$ denotes users who rated item $\boldsymbol{i}$. Sometimes a set denoted by $\boldsymbol{N(u)}$, is used, which contains all the items for which $\boldsymbol{u}$ provided an implicit preference like items that user rented, purchased, watched, etc. Models for the rating data are learnt by fitting the previously observed ratings. However, the goal is to generalize those in a way that allows predicting future unknown ratings, but caution should be exercised to avoid "over fitting" the observed data, this could be achieved by regularizing the learnt parameters using constants $\boldsymbol{\lambda_1, \lambda_2, \ldots \lambda_n}$. Exact values of these constants are determined by cross validation. Average overall rating is denoted by $\boldsymbol{\mu}$. A prediction for an unknown rating $\boldsymbol{r_{ui}}$ is denoted by $\boldsymbol{b_{ui}}$:

$$b_{ui} = \mu + b_u + b_i$$

The parameters b_u and b_i indicate the observed deviations of user u and item i, from the average. For example, suppose that we want to know a rating of the movie "The Terminator" by some user John. Let us say that the average rating over all movies, μ, is 3 stars. "The Terminator" is a popular movie, so it tends to be rated 0.6 stars above the average. On the other hand, John is a critical user, who tends to rate movies 0.4 stars lower than the average. Thus, the baseline predictor for "The Terminator" rating by John would be 3.2 stars by calculating $3-0.4+0.6$. In order to estimate b_u and b_i the least squares problem should be solved (2.1) [Koren and Bell, 2011]

$$min_b = \sum_{(u,i)\in K}(r_{ui} - \mu - b_u - b_i)^2 + \lambda_1(\sum_u b_u^2 + \sum_i b_i^2) \qquad (2.1)$$

Here, the term $\sum_{(u,i)\in K}(r_{ui} - \mu - b_u - b_i)^2$ strives to find b_u's and b_i's that fit the given ratings. The regularizing term $\lambda_1(\sum_u b_u^2 + \sum_i b_i^2)$ – avoids over fitting by penalizing the magnitudes of the parameters. This least square problem can be solved "*fairly*" efficiently by the method of stochastic gradient descent, e.g. for the Netflix database the mean rating μ is 3.6. As for the learned user biases b_u, their average is 0.044 with standard deviation of 0.41. The average of their absolute values $|b_u|$ is: 0.32. The learned item biases b_i average to –0.26 with a standard deviation of 0.48. The average of their absolute values $|b_i|$ is 0.43. An easier, yet somewhat less accurate way to estimate these parameters is by decoupling the calculation of the b_i's from the calculation of the b_u's. First, for each item i we set:

$$b_i = \frac{\sum_{u\in R(i)}(r_{ui} - \mu)}{\lambda_2 + |R(i)|}$$

Then, for each user u we set

$$b_u = \frac{\sum_{i\in R(u)}(r_{ui} - \mu - b_i)}{\lambda_3 + |R(u)|}$$

Averages are shrunk towards zero by using the regularization parameters **λ2, λ3**, which are determined by cross validation. Typical values for the Netflix dataset are: **λ2 = 25, λ3 = 10.**

In previous paragraph was considered an explicit user feedback, when it is unavailable, there are always additional sources of implicit feedback, and they can be exploited for better

understanding user's behavior. This helps to combat data *sparseness* and can be particularly helpful for users with few *explicit ratings*. Another well-known recommender system – the Amazon uses *browsing* and *purchase history* to get additional *implicit feedback* of the each user. In the Netflix dataset, the perfect source of the implicit feedback could be movie rental history, which directly shows user's preferences without requiring to rate rented movies. For example, the Netflix does not only tell us the rating values, but it also tells which movies users rate, regardless of how they rated these movies. In other words, a user *implicitl*y tells about his/her preferences by choosing to voice her opinion and vote by high or low rating. This creates a *binary matrix*, where "1" stands for "rated", and "0" for "not rated". While this binary data may not be as informative as other independent sources of implicit feedback, incorporating this *implicit data* does significantly improves prediction accuracy. The benefit of using the binary data is closely related to the fact that ratings are not missing at random, and users deliberately choose which items to rate [Marlin *et al.,* 2007].

2.1.1 – Matrix factorization methods applicable to recommendations

Latent factor models used in collaborative filtering to uncover latent features that explain received ratings, also known as SVD–based (singular vector decomposition) models. Matrix factorization models have gained popularity, because of their relatively high accuracy and scalability. The complications of usage SVD with explicit ratings in the collaborative filtering domain are the high amount of missing data (not rated items), SVD model is *undefined* when knowledge about the matrix is incomplete and prone to overfitting. Past works in matrix factorization relied on *imputation* [Kim and Yum, 2005], which fills in missing ratings and makes the rating matrix "*dense*". However, imputation can be very expensive as it significantly increases the amount of data. In addition, the data may be considerably "*distorted*" due to inaccurate imputation. Hence, other works [Bell *et al.,* 2007] suggested modeling of only explicit ratings, while avoiding overfitting through an adequately regulated model.

2.1.1.1 – SVD model

Let us consider what was shown about the SVD model in the article "Advances in collaborative filtering" [Koren and Bell, 2011]. SVD model maps both users and items to a joint latent factor space of dimensionality ***f***, such that user–item interactions are modeled as inner products in that space. The latent space tries to explain ratings by characterizing both items and users on factors automatically inferred from user feedback. For the case with WARES where items are different web analytics tools, ratings could be calculated as an average of several inferior ratings, representing certain web analytic tool features, e.g. *user statistics, software platforms, geo statistics, traffic statistics*, etc. Each of the inferior rating is measured by factors, e.g. for *user statistics*, it would have following dimensions: user type, count of sessions, days since last session, users, page views, unique page viewers, new users, number of sessions per user, days active. Accordingly, each item ***i*** is associated with a vector $\boldsymbol{q_i \in \mathbf{R}^f}$, and each user, if there are many, ***u*** is associated with a vector $\boldsymbol{p_u \in \mathbf{R}^f}$. For a given item ***i***, the elements of $\boldsymbol{q_i}$ measure the extent to which the item possesses those factors, positive or negative. For a given user ***u***, the elements of $\boldsymbol{p_u}$ measure the extent of interest the user has in items that are high on the corresponding factors, which could be positive or negative.

Dot product between two vectors $\boldsymbol{x,y \in \mathbf{R}^f}$ is $x^T y = \sum_{k=1}^{\int} x_k * y_k$ (2.2)

The resulting dot product $q_i^T p_u$ (2.2) captures the interaction between the user ***u*** and item ***i***, the overall interest of the user in characteristics of the item. The final rating is created by adding aforementioned baseline predictors that depend only on the user or item. Thus, a rating is predicted by the following rule:

$$\widehat{r_{ui}} = \mu + b_i + b_u + q_i^T p_u \qquad (2.3)$$

In order to learn the model parameters ($\boldsymbol{b_u}$, $\boldsymbol{b_i}$, $\boldsymbol{p_u}$ and $\boldsymbol{q_i}$) we minimize the regularized squared error:

$$\min_{b_*, q_*, p_*} \sum_{(u,i) \in K} (r_{ui} - \mu - b_i - b_u - q_i^T p_u)^2 + \lambda_4 (b_i^2 + b_u^2 + \left|\left|q_i\right|\right|^2 + ||p_u||^2)$$

The constant $\boldsymbol{\lambda_4}$, which controls the extent of regularization, is usually determined by cross validation. Minimization is typically performed by either stochastic gradient descent or

alternating least squares. Alternating least squares techniques rotate between fixing the p_u's to solve for the q_i's and fixing the q_i's to solve for the p_u's. Notice that when one of these is taken as a constant, the optimization problem is quadratic and can be optimally solved as seen in [Bell and Koren, 2007], [Bell *et al.,* 2007]. An easy stochastic gradient descent optimization was popularized by [Funk, 2016]. The algorithm loops through all ratings in the training data. For each given rating r_{ui}, a prediction $\hat{r}_{ui}$ is made, and the associated prediction error $e_{ui} = r_{ui} - \hat{r}_{ui}$ is computed. For example in some training case r_{ui}, we could modify parameters by moving in the opposite direction of the gradient, yielding:

$b_u \leftarrow b_u + \gamma*(e_{ui} - \lambda_4*b_u)$
$b_i \leftarrow b_i + \gamma*(e_{ui} - \lambda_4*b_i)$
$q_i \leftarrow q_i + \gamma*(e_{ui}*p_u - \lambda 4*q_i)$
$p_u \leftarrow p_u + \gamma*(e_{ui}*q_i - \lambda*p_u)$

Where γ – learning rate, and λ_4 regularization parameter. Better accuracy could be achieved by dedicating separate learning rates γ and regularization λ to each type of learned parameter. Thus, it is advised to employ distinct learning rates to user biases, item biases and the factors themselves. A good, intensive use of such a strategy is described in [Takacs *et al.,* 2008].

2.1.1.2 – SVD++ model

SVD++ allow us to increase accuracy of original SVD by considering an implicit feedback, which provides an additional indication of user's preferences [Koren, 2008]. This is especially helpful for those users that provided much more implicit feedback than explicit one. To this end, a second set of item factors is added, relating each item i to a factor vector $y_i \in \mathbf{R}^f$. Those new item factors are used to characterize users based on the set of items that this user has been rated, this is such a called SVD++ model:

$$\widehat{r_{ui}} = \mu + b_i + b_u + q_i^T (p_u + |R(u)|^{-\frac{1}{2}} * \sum_{j \in R(u)} y_j) \qquad (2.4)$$

The set $\mathbf{R}(u)$ contains the items rated by user u. Now, a user u is modeled as $p_u + |R(u)|^{-\frac{1}{2}} * \sum_{j \in R(u)} y_j$. Here a free user–factors vector, p_u is used much like in (2.3), which is learnt from the given explicit ratings. This vector is complemented by the sum $p_u +$

$|R(u)|^{-\frac{1}{2}} * \sum_{j \in R(u)} y_j$, which represents the perspective of implicit feedback. Since the y_j**'s** are centered around zero, by the regularization, the sum is normalized by $|R(u)|^{-\frac{1}{2}}$, in order to stabilize its variance across the range of observed values of $|\boldsymbol{R(u)}|$. Model parameters are determined by minimizing the associated regularized squared error function through stochastic gradient descent.

2.1.1.3 – SVDtime model

Finally, the most accurate and most sophisticated model for the matrix factorization is the SVDtime model, it concludes the latent factor models review. Above, the temporal dynamics was omitted, but users could change their preferences over time. An example of using SVDtime for WARES, a company owner who is using PIWIK and stores all website statistics on his own server, who does not want to share his/her data with others while using third party cloud-based solutions, like Google Analytics, next year could switch to cloud-based services due to lack of money needed for supporting own IT department. This type of evolution is modeled by taking the user factors, vector $\boldsymbol{p_u}$, as a function of time. In fact, these temporal effects are the hardest to capture, once again, we need to model those changes at the very fine level of a daily basis, while facing the built–in scarcity of user ratings. The resulting model is denoted as *SVDtime* where ratings predicted as:

$$\widehat{r_{ui}} = \mu + b_i(t_{ui}) + b_u(t_{ui}) + q_i^T(p_u(t_{ui}) + |R(u)|^{-\frac{1}{2}} * \sum_{j \in R(u)} y_j \ ) \qquad (2.5)$$

Where $\mathbf{b_i(t_{ui})}$ is a time–changing item biases, $\boldsymbol{b_u(t_{ui})}$ are real valued functions that changes over time and user's preferences are $\boldsymbol{p_u(t_{ui})}$. Time complexity per iteration is linear with the input size, while running time is approximately doubled compared to SVD++, due to the extra overhead required for updating the temporal parameters. Importantly, convergence rate was not affected by the temporal parameterization, and the process converges in around 30 iterations [Koren and Bell, 2011].

2.1.2 – Neighborhood methods applicable to recommendations

This is the most common approach in collaborative filtering, based on user-user neighborhood models [Herlocker *et al.,* 1999], where estimating unknown ratings based on recorded ratings of users with the same opinion. Later, appeared a similar approach for items called the item–item [Linden *et al.,* 2003]. A bright example of using neighborhood models is the web site Amazon.com, which became the largest internet-based retailer. In neighborhood methods, a rating is estimated using known ratings made by the same user on similar items. Item-item approach have better scalability and accuracy in many cases [Bell and Koren, 2007]. In addition, item–item methods are more capable in explaining the reasoning behind the ratings predictions, because users are familiar with items they have rated before.

This section will be focused on the item–item approaches and will use some explanations about considered methods from the article of Yehuda Koren and Robert Bell [Koren and Bell, 2011], of course in the context of the WARES web analytics tool will be considered as an item. Comparing to the previous section, latent factor models have more possibilities in representation of different aspects of the data and provides more accurate results than neighborhood methods, but neighborhood models dominates in e-commerce solutions mostly due to their simplicity, and ability to provide intuitive explanations of the reasoning behind given recommendations, e.g. the Amazon uses well-known "What other items do customers buy after viewing this item…", this aspect most ordinary users values more than accuracy. Second, they can provide immediate recommendations based on newly entered user's feedback, their main advantages are:

- **Simplicity**: neighborhood–based methods are relatively simple to implement. In its simplest form, only one parameter, the number of neighbors is used during the prediction process.
- **Justifiability**: as was mentioned before, neighborhood methods provides a *readable* and *comprehensible justification* for the computed predictions e.g. the list of items, as well as the ratings given by the user to these items, which can help a user better understand given recommendation and its relevance [Bell *et al.,* 2007].

- **Efficiency**: neighborhood-based methods require no costly training phases, ratings could be done when users are offline, thus giving faster whole recommendations when connected to the online database, memory requirements are minimal comparing to any other methods, making them scalable for any large applications.
- **Stability**: final dataset almost unaffected by addition of new users and items, meaning that when all similarities have been calculated there is no need to recalculate everything with few new users/items, only similarities between this new item and the ones already existing in the system need to be recalculated.

The main goal of the *neighborhood methods* is the same as for the matrix factorization approach: to find the best product/item ***i*** for user ***u*** and how to give him ***top–N*** recommendations, and do it in the most accurate way. Let us consider *item–item* rating calculation problem from the point of view of neighborhood methods [Koren and Bell, 2011]. We have ***I*** – set of all items and ***Iu*** – a set of items rated to a user ***u***, than the first problem is finding item ***i∈I/Iu*** for which **u** is most likely to be interested in. When ratings are available, this task is most often defined as a *regression* or (multi–class) classification problem, where the goal is to learn a function ***f:U×I→S*** that predicts a rating ***f(u,i)*** of a user ***u*** for a new item ***i***. Where ***U*** is a set of all users and **S** is the set of possible values for ratings. This function than will be used for recommending to a currently active user u_a an item i^* with the rating, which has the highest estimate value.

$$i^* = \arg\max f(u_a, j) \qquad j \in \frac{I}{I_u} \tag{2.6}$$

To be able to evaluate the performance of the given recommendation the accuracy could be evaluated, to do this ratings ***R*** are divided into a training set R_{tr} used to learn function ***f***, and a test set R_t is used to evaluate accuracy of the prediction, and then mean absolute error, MAE – 2.7, or root mean squared error, RMSE – 2.8, function is applied.

$$MAE(f) = \frac{1}{|R_t|}\sum_{R_{ui}\in R_t} |f(u,i) - r_{ui}| \tag{2.7}$$

$$RMSE\ (f) = \sqrt{\frac{1}{|R_t|}\sum_{R_{ui}\in R_t}(f(u,i) - r_{ui})^2} \tag{2.8}$$

For the WARES starting ratings are not available, because the designed system intend to make recommendations for a *single new user.* In this case the problem of finding the best suitable web analytic tools is usually transformed into the task of recommending to an active user u_a a list L_{ua} containing N items likely to interest him/her, or such a called "top list" [Crestani and Lee, 2000]. The quality of such method can be evaluated by splitting the items of I into a training set I_{tr}, used to learn function L, and a test set I_t. Let us say the set of items that user find relevant is $T(u) \subset I_u \cap I_t$ if the user responses are binary (yes or no), these can be the items that u has rated positively. Otherwise, if only a list of purchased or accessed items is given for each user u, then these items can be used as $T(u)$. The performance of this method is then computed using the measures of precision (2.9) and recall (2.10).

$$Prescision(L) = \frac{1}{|U|}\sum_{u \in U} \frac{L(u) \cap T(u)}{L(u)} \qquad (2.9)$$

$$Recall(L) = \frac{1}{|U|}\sum_{u \in U} \frac{L(u) \cap T(u)}{T(u)} \qquad (2.10)$$

But the drawback of this recommendation method is that the all items from the list $L(u)$ are considered *equally "interesting"* to a user u and thus the final decision is made by a user, not by a recommender system, and he/she need to sort this results on his own.

2.1.2.1 – Rating prediction and classification using standard Neighborhood methods

Regression user–based neighborhood recommendation method [Koren and Bell, 2011], tends to predict rating r_{ui} of a user u for a new item i using ratings given to i by users most similar to u. Suppose we have for each user $v \neq u$ a value w_{uv} representing the preference similarity between u and v, how this similarity can be computed was described in the previous section. The *k–nearest–neighbors*, next ***k*–NN**, of u, denoted by $N(u)$, are the k users v with the highest similarity w_{uv} to u. However, only the users who have rated item i can be used in the prediction of r_{ui}, and thus we should consider the k users most similar to u that have rated I, let this set of neighbors be $Ni(u)$, then the rating r_{ui} can be estimated as the average rating given to i by these neighbors:

$$\widehat{r_{ui}} = \frac{1}{|N_i(u)|}\sum_{v \in N_i(u)} r_{vi} \qquad (2.13)$$

A problem with 2.13 is that is does not take into account the fact that the neighbors can have different levels of similarity. A common solution to this problem is to weigh the contribution of each neighbor by its similarity to $\boldsymbol{u}$. However, if these weights do not sum to 1, the predicted ratings can be well outside the range of allowed values. Consequently, it is customary to normalize these weights, such that the predicted rating becomes:

$$\widehat{r_{ui}} = \frac{\sum_{v \in N_i(u)} w_{uv} * r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|} \qquad (2.14)$$

Where $\boldsymbol{w_{uv}}$ is a user's weight contribution $|\boldsymbol{w_{uv}}|$ is used instead of $\boldsymbol{w_{uv}}$ because negative weights can produce ratings outside the allowed range. Also, $\boldsymbol{w_{uv}}$ can be replaced by w_{uv}^{α} where $\alpha > 0$ is an amplification factor [Breese *et al.,* 1998] When $\alpha > 1$, as is it most often employed, an even greater importance is given to the neighbors that are the closest to $\boldsymbol{u}$. Equation (2.23) also has an important flaw: it does not consider the fact that users may use different rating values to quantify the same level of appreciation for an item. For example, one user may give the highest rating value to only a few outstanding items, while a less difficult one may give this value to most of the items he likes. This problem is usually addressed by converting the neighbors ratings $\boldsymbol{r_{vi}}$ to normalized ones $\boldsymbol{h(r_{vi})}$ [Breese *et al.,* 1998] and giving the following prediction:

$$\widehat{r_{ui}} = h^{-1} * \frac{\sum_{v \in N_i(u)} w_{uv} * h(r_{vi})}{\sum_{v \in N_i(u)} |w_{uv}|}$$

Note that the predicted rating must be converted back to the original scale, hence the $\boldsymbol{h^{-1}}$ in the equation. So when it comes to assigning a rating to an item, each user has its own personal scale. Even if an explicit definition of each of the possible ratings is supplied (e.g. 1="strongly disagree", 2="disagree", 3="neutral", etc.), some users might be reluctant to give high/low scores to items they liked/disliked. The most common approaches to normalize are *Mean–centering* and *Z–score*, described below.

Mean–centering. The idea of mean–centering [Breese *et al.,* 1998] is to determine whether a rating is positive or negative by comparing it to the mean rating. In *user–based*

recommendation, a raw rating r_{ui} is transformation to a mean–centered one $h(r_{ui})$ by subtracting to r_{ui} the average r_u of the ratings given by user u to the items in I_u:

$$h(r_{ui}) = r_{ui} - r_u$$

Using this approach the user–based prediction of a rating r_{ui} is obtained as:

$$\widehat{r_{ui}} = r_u + \frac{\sum_{v \in N_i(u)} w_{uv} * (r_{vi} - r_v)}{\sum_{v \in N_i(u)} |w_{uv}|}$$

In the same way, the item–mean–centered normalization of r_{ui} is given by:

$$h(r_{ui}) = r_{ui} - r_i$$

Where $\hat{r}_i$ corresponds to the mean rating given to item i by user in U_i. Rating r_{ui} is predicted as:

$$\widehat{r_{ui}} = r_u + \frac{\sum_{j \in N_u(i)} w_{ij} * (r_{uj} - r_j)}{\sum_{j \in N_u(i)} |w_{ij}|}$$

This normalization technique is most often used in *item–based* recommendation, e.g. on Amazon.com. An interesting property of mean–centering is that one can see right–away if the appreciation of a user for an item is positive or negative by looking at the sign of the normalized rating. Moreover, the module of this rating gives the level at which the user likes or dislikes the item.

Z–score, two users A and B that both have an average rating of 3. Moreover, suppose that the ratings of A alternate between 1 and 5, while those of B are always 3. A rating of 5 given to an item by B is more exceptional than the same rating given by A, and, thus, reflects a greater appreciation for this item. While *mean–centering* removes the offsets caused by the different perceptions of an average rating, Z–score normalization [Herlocker *et al.,* 1999] also considers the spread in the individual rating scales. Once again, this is usually done differently in *user–based* than in *item–based* recommendation. In *user–based* methods, the normalization of a rating r_{ui} divides the *user–mean–centered* rating by the standard deviation σ_u of the ratings given by user u:

$$h(r_{ui}) = \frac{r_{ui} - r_u}{\sigma_u}$$

A *user–based* prediction of rating r_{ui} using this normalization approach would therefore be obtained as:

$$\widehat{r_{ui}} = r_u + \sigma_u * \frac{\sum_{v \in N_i(u)} w_{uv} * (r_{vi} - r_v)/\sigma_u}{\sum_{v \in N_i(u)} |w_{uv}|}$$

Likewise, the Z–score normalization of r_{ui} in item–based methods divides the *item–mean–centered* rating by the standard deviation of ratings given to item i:

$$h(r_{ui}) = \frac{r_{ui} - r_i}{\sigma_u}$$

The item–based prediction of rating r_{ui} would then be:

$$\widehat{r_{ui}} = r_u + \sigma_u * \frac{\sum_{j \in N_u(i)} w_{ij} * (r_{vj} - r_j)/\sigma_u}{\sum_{j \in N_{ui}(i)} |w_{ijuv}|}$$

In some cases, rating normalization can have undesirable effects. For instance, imagine the case of a user that gave only the highest ratings to the items he has purchased. Mean–centering would consider this user as "easy to please" and any rating below this highest rating, whether it is a positive or negative rating, would be considered as negative. However, it is possible that this user is in fact "hard to please" and carefully selects only items that he will like for sure.

To answer the question which implementation (*user–based* or an *item–based*) will suit the most recommender system's goals, let us consider table 2 and 3, from the article "Advances in collaborative filtering" [Koren and Bell, 2011]:

Table 2: The average number of neighbors vs. average number of ratings

	# of Avg. neighbors	**# of Avg. ratings**
User–based method	$(\lvert U\rvert - 1)\left(1 - \left(\frac{\lvert I\rvert - p}{\lvert I\rvert}\right)^p\right)$	$\frac{p^2}{\lvert I\rvert}$
Item–based method	$(\lvert I\rvert - 1)\left(1 - \left(\frac{\lvert U\rvert - q}{\lvert U\rvert}\right)^q\right)$	$\frac{q^2}{\lvert U\rvert}$

Uniform distribution of ratings is assumed with average number of ratings per user $p = |R|/|U|$, and average number of ratings per item $\boldsymbol{q} = |\boldsymbol{R}|/|\boldsymbol{I}|$

Table 3: The space and time complexity of user–based and item–based neighborhood

	space	Time training	Time online
User–based method	$O(\|U\|^2)$	$O(\|U\|^2 * p)$	$O(\|U\| * k)$
Item–based method	$O(\|I\|^2)$	$O(\|I\|^2 * q)$	$O(\|I\| * k)$

Where $\boldsymbol{p} = \boldsymbol{max_u}\ |\boldsymbol{I_u}|$, is the maximum number of ratings per item $\boldsymbol{q} = \boldsymbol{max_i}\ |\boldsymbol{U_i}|$, and $\boldsymbol{k}$ is the maximum number of neighbors used during the prediction process. For the WARES recommender system, it is clear that we should use item-item approach, because by its nature it relies on items rather than ratings made by numerous users.

2.1.3 – Typical similarity measurement methods in recommender systems

Measure of the *similarity* between two objects $\boldsymbol{a}$ and $\boldsymbol{b}$, is often used in *information retrieval*, it consists in representing these objects in the form of two vectors $\mathbf{x}_{\boldsymbol{a}}$ and $\mathbf{x}_{\boldsymbol{b}}$ and computing the *Cosine Vector* or *Vector Space similarity* [Billsus *et al.,* 2002] between these vectors:

$$\cos(x_a, x_b) = \frac{x_a * x_b}{||x_a|| * ||x_b||}$$

In the context of item recommendation, this measure can be employed to compute user's similarities by considering a user $\boldsymbol{u}$ as a vector $\mathbf{x}_{\boldsymbol{u}} \in \mathbf{R}^{|I|}$, where $\mathbf{x}_{\boldsymbol{ui}} = \boldsymbol{r_{ui}}$ if user $\boldsymbol{u}$ has rated item $\boldsymbol{i}$, and 0 otherwise. The similarity between two users $\boldsymbol{u}$ and $\boldsymbol{v}$ would be computed as:

$$\text{Cosine Vector}(\text{u}, \text{v}) = \cos(x_u, x_v) = \frac{\sum_{i \in I_{uv}} r_{ui} * r_{vi}}{\sqrt{\sum_{i \in I_u} r_{ui}^2 * \sum_{j \in I_v} r_{vj}^2}}$$

Where $\boldsymbol{I_{uv}}$ once denotes the items rated by both $\boldsymbol{u}$ and $\boldsymbol{v}$. A problem with this measure is that it does not consider the differences in the mean and variance of the ratings made by users $\boldsymbol{u}$ and $\boldsymbol{v}$. but Pearson Coefficient does, see below.

A popular measure of similarity in the most of "*item–item*" approaches is based on the Pearson correlation coefficient, ρ_{ij}, which measures the tendency of users to rate items ***i*** and ***j*** similarly. Since many ratings are unknown, some items may share only a handful of common observed raters. The empirical correlation coefficient, ρ_{ij}, based only on the common user's support. It is advised to work with residuals from the baseline predictors, b_{ui}, to compensate for *user–specified* and *item–specific* deviations. Thus, the approximated correlation coefficient is denoted by the following formula:

$$p_{ij} = \frac{\sum_{u \in U(i,j)} (r_{ui} - b_{ui}) * (r_{uj} - b_{uj})}{\sqrt{\sum_{u \in U(i,j)} (r_{ui} - b_{ui})^2 * \sum_{u \in U(i,j)} (r_{uj} - b_{uj})^2}} \qquad (2.11)$$

The set ***U(i, j)*** contains users who rated both items ***i*** and ***j***. Because estimated correlations based on a greater user support are more reliable, an appropriate similarity measure, denoted by s_{ij}, is a shrunk correlation coefficient of the following form:

$$s_{i,j} = \frac{n_{i,j} - 1}{n_{i,j} - 1 - \lambda_8} \qquad (2.12)$$

The variable $n_{ij} = |U(i, j)|$ denotes the number of users that rated both ***i*** and ***j***. A typical value for λ_8 is 100. Such shrinkage can be motivated from a Bayesian perspective see [Gelman *et al.,* 1995].

2.2 – Content–based filtering approach and possible ways of applications in WARES.

Content–based recommender systems, distinct from the Collaborative–based recommenders in the way that for rating calculation they *matching* different attributes of *user's profile*, with the attributes of some "content" object, instead of matching ratings given by the different users. This section provides an overview of the content–based filtering methods, their advantages and drawbacks and some algorithms used for the rating prediction.

To implement a content–based recommendation approach, a set of documents and/or descriptions of items previously rated by a user should be analyzed, and to build a model or

profile of user's interests based on the features of the objects rated by that user. A typical user profile suitable for content-based filtering consists of a structured representation of user's interests, adopted to recommend new "interesting" items. The recommendation process works as continuous process of matching attributes of the user's profile against the attributes of the content object. The result is a relevance judgment that represents the user's level of interest in that object. In context of the WARES recommender system, content-based techniques could be used to filter online search results for the web analytic tools and grouping them according to the original user's preferences.

Most research on content–based recommender systems takes place at the intersection of Information Retrieval [Baeza–Yates and Ribeiro–Neto, 1999] and Artificial Intelligence. From information retrieval, research on recommendation technologies derives the vision that users searching for recommendations are engaged in an information seeking process. In information retrieval systems the user expresses a one–off information need by giving a query, which usually a list of keywords. Items to be recommended can be very different depending on the number and types of attributes used to describe them. Each item can be described through the same small number of attributes with known set of values, in the case of WARES this is not an appropriate form for Web Analytic tools, because in all cases they are being described through unstructured text. In that case there are no attributes with well–defined values, thus usage of document modeling techniques from the domain of information retrieval is required, see section 2.2.2 for details.

2.2.1 – Advantages and disadvantages of the content–based filtering

Content-based filtering approach has strong and weak sides comparing to collaborative filtering, let us consider how these sides was described by Google staff research scientist, PhD in computer science Yehuda Koren [Koren and Bell, 2011]:

- **User independence**: content–based systems uses only ratings provided by the active user in order to build its own profile, while collaborative-based systems requiring lots of ratings from other users in order to give more or less suitable recommendations.

- **Transparency**: explicit explanations is a strong side of the content-based methods, e.g. a simple listing on which recommendations was based could be very helpful for any user in order to decide whether to trust a recommendation or not, while most collaborative-based systems considered as a "black boxes" where is no explanation how recommendation was built, except "someone else also liked it".
- **New items:** content–based recommenders can recommend items which are not yet being rated. Thus they almost do not suffered from such a called "cold-start" problem for recommending items, while still have problems with "user cold-start", which also affects most collaborative-based systems.

This advantages seems very impressive, but there also exists some major drawbacks:

- **Limited context analysis:** for the content–based techniques domain knowledge is often needed in order to produce reliable recommendations e.g., WARES recommendations system needs to know as many properties as possible for an entity "web analytic tool", and that is why usage of the domain ontology is strongly recommended, see section 3.3 for the details.
- **Over–specialization:** content–based recommender systems does not possess methods for inferring some unexpected results. Meaning the system cannot recommends items which features goes beyond user's profile settings, meaning the user will most likely get items similar to those he/she already rated. This drawback is also called a *serendipity problem*.
- **New user problem:** in order to produce "accurate" recommendations for a particular user recommender system need to collect enough ratings to "understand" user's preferences. Meaning, when few ratings are available, as for a new user, the system will not be able to provide accurate recommendations for the current user, a "*user cold start*" problem. Some strategies for tackling the "*cold start*" problem will be considered later in the sub-section 2.2.3.

2.2.2 – Item representation in the content–based recommender approach

In the content-based methods, all items from the dataset are described by a certain number of features, called attributes or properties. For example in the WARES recommender system features describing a certain web analytic tool, next item, are: *unique number of visitors, clicks, price, user's interface language, special unique feature*s, etc... When each item is described by the same set of attributes, and there is a known set of values the attributes may take, the items are represented as a *structured* data. In this case, many *machine learning* algorithms can be used in order to learn a user's profile [Pizzani and Billsus, 2007]. In most content–based filtering systems, as in WARES recommender system, item descriptions consists of the features, which should be extracted from the web pages describing certain web analytic tools. For WARES we have unstructured data, thus attributes should be extracted before recommender process starts. Textual features create a number of complications when learning a user's profile, due to the natural language ambiguity. The problem is that traditional *keyword–based* profiles are unable to capture the semantics of user's interests because they are primarily driven by a string matching operation. If a string, or some morphological variant, is found, in both the profile and the document, a match is made and the document is considered as relevant, while string matching suffers from the following problems:

- **Polysemy:** when one word have multiple meanings.
- **Synonymy:** multiple words with the same meaning.

Thanks to *synonymy*, relevant information contained on the developer's websites, representing web analytic tool can be missed, while due to *polysemy*, wrong documents (websites) could be deemed relevant. Here the Semantic analysis comes in handy and its integration in personalization models is one of the most innovative and interesting approaches proposed in literature to solve those problems. The key idea is the adoption of knowledge bases, such as *lexicons* or *ontologies*, for annotating items.

In the "traditional" keyword–based *Vector Space Model* of the text document [Koren and Bell, 2011] keywords are represented by the frequency and inverse document frequency weightings, and each document is represented by a vector in the ***n***–dimensional space, where each dimension corresponds to a term from the overall vocabulary of a given document

collection. Formally, every document is represented as a vector of term weights, where each weight indicates the degree of association between the document and the term. Let $\boldsymbol{D = \{d_1, d_2, ..., d_N\}}$ denote a set of documents or corpus, and $\boldsymbol{T = \{t_1, t_2, ..., t_n\}}$ be the dictionary, that is to say the set of words in the corpus, $\boldsymbol{T}$ is obtained by applying some standard natural language processing operations, such as tokenization, stop–words removal, and stemming [Baeza–Yates and Ribeiro–Neto, 1999]. Each document $\boldsymbol{d_j}$ is represented as a vector in a *n*–dimensional vector space, so $\boldsymbol{d_j = \{w_{1j}, w_{2j}, ..., d_{nj}\}}$, where $\boldsymbol{w_{kj}}$ is the weight for term $\boldsymbol{t_k}$ in document $\boldsymbol{d_j}$.

This document representation raises two issues: weighting the terms and measuring the feature vector similarity. The most commonly used term weighting scheme is *Term Frequency–Inverse Document Frequency* (TF–IDF) weighting is based on following empirical observations regarding text [Salton, 1989]:

- **Rare terms** are not less relevant than frequent terms – IDF assumption.
- **Multiple occurrences** of a term in a document are not less relevant than single occurrences, TF assumption.
- **Long documents** are not preferred to short documents (normalization assumption).

In other words, terms that occur frequently, TF, in one document, for the WARES document are replaced with websites, but rarely in the rest of the corpus, IDF, are more likely to be relevant to the topic of the document. These assumptions represented in the TF–IDF function:

$$TF - IDF(t_k, d_j) \quad = TF(t_k, d_j) * \log \frac{N}{n_k} \qquad (2.15)$$

Where $\boldsymbol{N}$ denotes the number of documents in the corpus, and $\boldsymbol{n_k}$ denotes the number of documents in the collection in which the term $\boldsymbol{t_k}$ occurs at least once, and $\log \frac{N}{n_k}$ is a formula for IDF.

$$TF(t_k, d_j) = \frac{f_{k,j}}{\max_z f_{z,j}}$$

The maximum is computed over the frequencies $\boldsymbol{f_{z,j}}$ of all terms $\boldsymbol{t_z}$ that occur in document $\boldsymbol{d_j}$. In order for the weights to fall in the $\{0, 1\}$ interval and for the documents to be represented by vectors of equal length, weights obtained by Equation (2.15) are usually normalized by *cosine* normalization, which enforces the normalization assumption:

$$w_{k,j} = \frac{TF - IDF(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|}(TF - IDF(t_s, d_j)^2}}$$

For example, suppose that we have term count tables of a corpus consisting of only two documents, shown on the figure 13.

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Figure 13: Two documents with terms

The calculation of TF–IDF for the term "this" is performed in its raw frequency form, TF is just the frequency of the "this" for each document. In each document, the word "this" appears once, but as the document 2 has more words, its relative frequency is smaller. An IDF is constant per corpus, and accounts for the ratio of documents that include the word "this". In this case, we have a corpus of two documents and all of them include the word "this". So TF–IDF is zero for the word "this", which implies that the word is not very informative as it appears in this documents.

The methods of similarity measures, used to describe the proximity of two vectors, was considered in the section 2.1.3. In the content–based recommender systems, relying on vector space model, both user profiles and items are represented as a weighted term vectors. For example, predictions of a user's interest in a particular item can be derived by computing the cosine similarity or Pearson correlation. But the most important lesson learned from the analysis of the recommender systems developed in the last years [Ricci *et al.,* 2015] is that *keyword–based* representation for both items and profiles can give accurate predictions. *Keyword–based* approach works pretty fine, but shows its *limitation*s connected to synonymy and polysemy. For example, in the WARES recommender system a user needs web analytic tool for analysis of user's "click stream" on his website, keyword–based approaches will only find websites in which the words "click" and "stream" occur. But websites with "clickstream and "click–stream"

will not appear in the set of recommendations, even though they are likely to be very relevant for that user. More advanced representation strategies are needed in order to equip content–based recommender systems with "semantic intelligence", see next section.

2.2.3 – Semantic analysis in the content–based recommender systems, using ontologies

Semantic analysis allows us creating more accurate profiles that contains descriptive data and concepts defined in the external knowledge bases. The main motivation for this approach is the challenge of providing a recommender system with the *linguistic background knowledge*, which characterizes the ability of interpreting natural language texts and reasoning on their content. The description of these strategies should be carried out by taking into account following criteria:

- **Type of knowledge source:** lexicon or ontology.
- **Techniques:** different techniques for the annotations and items representations.
- **Type of content** included in the user profile.
- **Item–profile** matching strategy.

Proposed recommender system, WARES, incorporates an ontology, created specifically for the domain of web analytic, of course it is not new implementation, several recommender systems created before also uses ontologies. As an example I would like to compare WARES with another recommender system for Interactive Digital Television, proposed by Blanco–Fernandez [Blanco–Fernandez *et al.,* 2008], where the authors apply reasoning techniques borrowed from the *Semantic Web* in order to compare user's preferences with items, in their case item are represented by TV programs. For this system the linguistic knowledge comes exclusively from the WordNet lexical ontology [WordNet]. TV programs available, during the recommendation process annotated by metadata that accurately describes their main attributes, both TV domain data and the user's profiles was created using Web Ontology Language [OWL, 2012]. The OWL languages are characterized by formal semantics, and are built upon a W3C XML [XML, 2008] standard for objects called the Resource Description Framework [RDF, 2014]. Ontology–profile provide a formal representation of the user's preferences, being able to

"reason" about them and "discover" extra knowledge about their interests. The recommendation phase exploits the knowledge stored in the user profile to discover hidden semantic associations between the user's preferences and the available products.

2.2.4 – Methods for Learning User's Profiles, applicable to the content–based recommender systems

For the task of inducing content–based profiles, *machine learning* techniques are widely used, and they are well–suited for text categorization [Sebastiani, 2002]. An inductive process automatically builds needed text classifier by learning from a set of "training documents" labeled with categories they belong to. For example, the problem of learning user profiles can be cast as a binary text categorization task: each document has to be classified as "relevant" or "not relevant" with respect to the user's preferences. Let us denote a set of categories by $\boldsymbol{C = \{c_+, c_-\}}$, where $\boldsymbol{c_+}$ is the positive class or "relevant" and $\boldsymbol{c_-}$ the negative or "not relevant" (user–dislikes). Below, in this section will be considered *machine learning* algorithms frequently used for the content–based recommender systems. They are able to learn a function that models user's interests. These methods typically require users to label documents by assigning a relevance score, and automatically infer profiles exploited in the filtering process to rank documents according to the user's preferences.

2.2.4.1 – Probabilistic Methods and Naïve Bayes

Naïve Bayes is a probabilistic approach to inductive learning, and belongs to the general class of *Bayesian classifiers*, it generates a probabilistic model based on previously observed data. The model estimates "a posteriori" probability, $\boldsymbol{P(c|d)}$, of the document $\boldsymbol{d}$ belonging to the class $\boldsymbol{c}$. This estimation is based on several probabilities: the a priori probability, $\boldsymbol{P(c)}$, of observing a document in the class $\boldsymbol{c}$, $\boldsymbol{P(d|c)}$, the probability of observing the document $\boldsymbol{d}$ given $\boldsymbol{c}$, and $\boldsymbol{P(d)}$, the probability of observing the instance $\boldsymbol{d}$. Using these probabilities, the Bayes theorem is applied to calculate $\boldsymbol{P(c|d)}$:

$$P(c|d) = \frac{P(c) * P(d|c)}{P(d)}$$

To classify the document ***d***, the class with the highest probability is being chosen:

$$c = argmax_{c_j} \frac{P(c_j) * P(d|c_j)}{P(d)}$$

Where ***P*(*d*)** is generally removed as it is equal for all c_j. As we do not know the value for ***P*(*d*|*c*)** and ***P*(*c*)**, we estimate them by observing the training data. Although naïve Bayes performance is not as good as some other learning methods such as nearest–neighbor classifiers or support vector machines, it has been shown that it can perform surprisingly well in the classification tasks [Domingos and Pizzani, 1997]. Another advantage of the naïve Bayes approach is that it is very efficient and easy to implement compared to other learning methods.

2.3 – Conclusion

In this chapter was surveyed possible ways and applications of the collaborative and content–based filtering techniques in recommender systems, was provided an extensive overview for the most important, in my opinion, algorithms, and techniques, which could be used for the WARES recommender system. Although there exist lot more algorithms and techniques for building different types of recommender systems, but I think it was necessary to consider what was considered because it explains why these methods should not be used in the WARES, while others should be. And yes there was used lots of citations from the external sources, of course accompanied by the references to the articles of these authors. Basics which was explained in this sections was written by the respected authors long before this paper was conceived, do not see the point explaining these basics from the scratch, while I can use them in the context of the WARES. It was discovered that both collaborative and content-based filtering approaches extensively uses user's profile information to better predict ratings, but differs in the way how existing user's ratings being used for creating item-item and user-item cross-ratings. In the section 2.1 was considered an issue when some items have not yet received their ratings, and how to overcome this issue using additional data from the implicit user's feedback sources. Was considered and explained how to use two well–known methods for measuring similarities between set of items: Pearson's correlation and cosine similarity. Finally, after considered content-based approach in the section 2.2, I came to the conclusion that for the WARES recommender system, some approaches, like collaborative filtering, and the set of algorithms connected to this technique are not applicable, due to the specificity of the recommended content. Collaborative filtering approach seems to be inappropriate because the WARES recommender system does not have great number of users, from the beginning it was conceived for a single user. Everything considered in this chapter was useful and helped me in the further implementation of the WARES recommender system.

Chapter 3 – Structure of the WARES recommender system

There exists lots of data processing and metadata processing methods, many user models, vast variety of filtering techniques, many accuracy metrics, and many levels of personalization. Due to its domain the WARES implies several constraints due to its purpose and specificity, e.g. the environment impose some architectural limitations; data may be not in a suitable format, number of users in the system, etc. This can make the task of building such a recommender system very complicated and a long–term task for a single person. In this chapter, relying on the studies of other works, will be shown how WARES was designed. Following existing guidings it will be shown how to build a solid architecture based on three models: data model, user model and application model.

3.1 – Generalized overview about recommender systems structure

After considering two well–known approaches for building recommender systems and become acquainted with corresponding algorithms and techniques it can be asserted that collaborative filtering approach does not suit WARES needs, because there is a major complication with collaborative approach, it needs users and their rating, lots of them, and WARES does not have this data, because it intends to give a recommendations to a single user, who supposed to use this recommender system for the first time that is why there is no thousands or even tens of users who can give rating on previously recommended web analytics tools. Collaborative techniques cannot be applied because WARES simply do not have data, as collaborative filtering recommender systems are based on the statistical processing of opinions expressed by *many* users. On the other hand the content–based approach seems more appropriate, because even with one user in the system, we can do *item–user* and *item-item* matching, using ontology, see section 3.3, as an extension to "items" (web analytic tool) profile. Summarizing what was learned about recommender systems, let us consider an existing example of the content-based recommender system proposed by the professor of the University of Bari in Italy, Pasquale Lops in his article "Content-based recommender systems: state of the art and trends" [Lops *et al.,* 2011] and examine which parts could be used for the WARES and which are not, see figure 15.

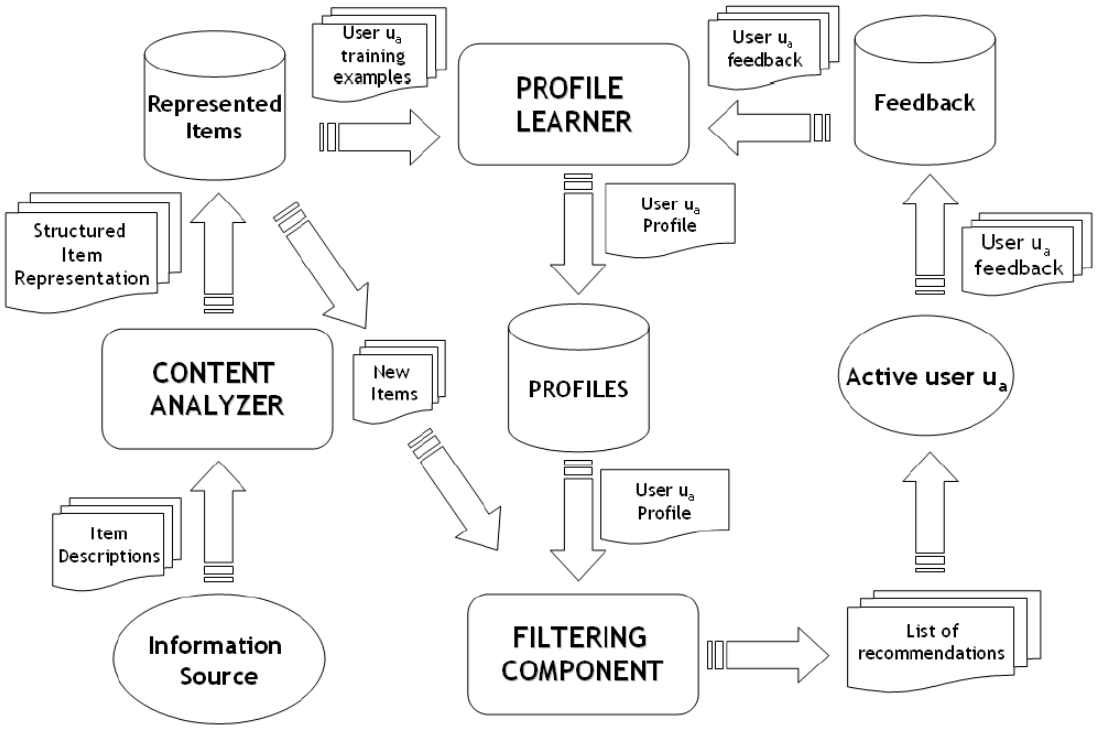


Figure 14: Architecture of the content–based recommender system, [Lops *et al.,* 2011]

In order to understand the main components of the content–based architecture let us explain each block. The recommendation process performed in three steps, each of which handled by a separate component:

- **Content analyzer** – When information has no structure, e.g. textual description of web analytic tool on the developer's website, some kind of pre–processing step is needed to extract structured relevant information. The main goal of this block – representation of the items, e.g. web pages containing valuable data about web analytics tool, in a form suitable for the profile learning.
- **Profile learner** – This module collects data representing user's preferences and tries to generalize this data, in order to construct a user profile. Usually, the generalization strategy implemented through the machine learning techniques, described in the section 2.2.4, which infers a model of the user's interests.
- **Filtering component** – This module exploits the user profile in order to suggest relevant items by matching the user's profile to items dataset. The result is a binary or continuous

item vectors, computed using one of the similarity metrics described in 2.1.3, in the latter case resulting a ranked list of potentially interesting items.

The first step of the recommendation process is the one performed by the *content analyzer*, which usually uses various information retrieval and data mining technique to gather relevant information useful for the recommendation process. Item descriptions coming from the "information source", for the WARES it is web pages containing description of the different web analytic tools, are processed by the *content analyzer*, which extracts features like *keywords, concepts*, etc. from the unstructured text to produce a structured item representation, stored in the repository *Represented Items*. In order to construct and update the *profile* of an *active user* feedbacks for items are stored in the dataset repository, later these feedbacks will be also used for ratings prediction. Typically, it is possible to distinguish between two kinds of relevance feedback: *positive*, inferring features liked by the user, and *negative*, inferring features the user does not like or not interested in. Two different techniques can be adopted for recording user's feedback: *explicit* and *implicit*, described through the sections 2.1.1 – 2.1.2.

To get an explicit feedback, while using content–based filtering approach, we can use one of the following well–known methods [Lops *et al.,* 2011]:

- **Like/dislike**: items are being classified as "relevant" or "not relevant" by adopting a simple binary rating scale 0, 1.
- **Ratings**: a discrete numeric scale is usually adopted to judge items.
- **Text comments**: comments about a single item are collected and presented to the users as a means of facilitating the decision–making process.

Now, when we considered the generalized architecture of the content-based recommender system and its components let us proceed to the design stage. The major concern at the stage of the design and planning that any further changes in the development is very costly. In order to tackle this problem in a systematic way, it is useful to step back and see from a wider perspective what are the main design decisions to make and the factors, which influence them. To facilitate the design process of the WARES recommender system I will follow the structural model for the recommender system proposed by the researcher from the Alcatel-Lucent Bell labs, Jerome Picault in his article "How to get the recommender out of the lab" [Picault *et al.,* 2011], the general model is shown on the figure 16.

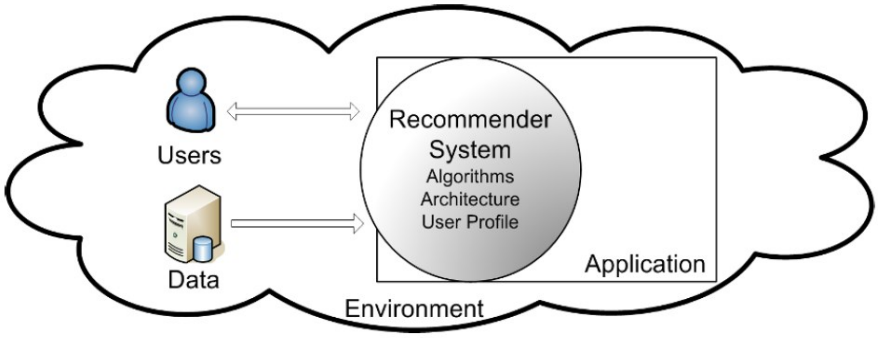


Figure 15: Recommender system in its environment [Picault *et al.,* 2011]

Designing a recommender system means making choices categorized into the three following domains:

- **Architecture**: how the system will be deployed, will it be centralized or distributed?
- **User's profile**: what is the user model, is profile adaptation needed?
- **Algorithms**: which recommendation methods to be used?

In most cases these choices are constrained by the environment of the recommender system. That is why when building a recommender system it is important to study system's environment. Following the best practice recommendations from the article of Jerome Picault [Picault *et al.,* 2011] I will describe the WARES' environment along three dimensions:

- **Application**: the overall scope of an application of which recommender is a part of?
- **Users**: who are the users, what are their goals?
- **Data**: what are the characteristics of the data on which recommendations are based?

3.2 – Understanding WARES recommender system environment

Let us consider in details these three models mentioned in the previous section and how they correspond to the WARES: the *user model, the data model, and the application model*. This section considers at large these models from the point of view described by Jerome Picault in his paper [Picault *et al.,* 2011], transferring proposed models to my recommender system will help me to decide how WARES should be implemented, and help to reveal key constraints of the WARES design, allowing to answer the following questions:

- Choice of the application model and recommendation algorithm for the WARES.
- Choice and possibilities of adaptation of the user's profile.
- Choice of the data model for the WARES recommender.

3.2.1 – Application model

Though a recommender system itself is a complex piece of software and as was mentioned before often is the part of a larger system (application). Globally the recommender system is one of the features of the application. It may be a minor feature or a main selling point, the application may be pre–existing or built together with the recommender, but in any case the design of the recommender system has to be integrated within the design of an application hosting it. This section studies the main factors regarding the host application influencing recommender system design, base features [Picault *et al.,* 2011] of the application model is shown in table 4.

Table 4: General application model components [Picault *et al.,* 2011]

Model feature	Possible values
Recommender purpose	major service, long–tail focused, increase revenues, increase loyalty, increase system efficiency
Recommender type	single item, multiple items, sequence
Integration with navigation features	Stand–alone application, or in–browser plugin
Performance criteria	correctness, transparency, serendipity, risk–taking, response speed, robustness to attack
Device to support the application	fixed, mobile, multiple
Number of users	Single or group of users
Application infrastructure	browser–based application, distributed application
Screen real–estate	limited, not limited

The table above contains all the possible values, they will be explained in the next two sub-sections, but not all of them are applicable to the WARES, those which are relevant is shown in table 5.

Table 5: Application model for the WARES recommender system

Recommender Purpose	major service - the only purpose of this system to give suggestions about "best" web analytics tool or set of tools, increase loyalty, increase revenues, increase system efficiency
Recommender type	multiple items
Integration with navigation features	"tight" integration through coupling with the information retrieval techniques to deliver a personalized search service
Performance criteria	First of all it is correctness than transparency and response speed
Device to support the application	desktop personal computers and laptops
Number of users	single use application
Application infrastructure	browser-based application with personalized search
Screen real-estate	not limited

As was mentioned above, the recommender system is always a part of an application, but is it just a small useful feature or the whole application is built around the recommender system? In the case with the WARES it is a major feature of an application, thus the whole application is built around the WARES.

3.2.1.1 – Understanding the recommender role in the application

The main question to answer before starting designing a recommender system application is to determine its main goals. It is not so easy as it seems, because depending on what was determined we could have long-lasting consequences through the future implementation stages. In this section will be considered all the possible features of the recommender system as an application. Relying on the list of properties from the table 4, as was proposed by the Jerome Picault in his article [Picault *et al.,* 2011], the WARES recommender system will have the following set features:

- **"Major service"** – RS may be a "major service" provided by the application. Many such recommender systems already have been developed in different fields, e.g. for music last.fm, for movies Netflix, as for the WARES, it was intended to be a "major service" from the beginning.
- **Increase loyalty of users**: customers usually return to the services that gives them best match their needs. Loyalty can be increased by involving users in the recommendation

process, asking them for ratings or manual profile, highlight new recommendations, etc. This is not the case for WARES, because I have no intention to implement any long–time stored user profiles, to make my system "safer", in a sense of personal data.

– **Increase revenues:** through the promotion of targeted products. In that case, the recommendations would be determined by both the user's preferences and some marketing rules defined to follow a particular strategy. It is necessary to carefully balance the expectations of the users and the business strategy, to ensure users perceive value in the system. It could be an option when WARES will be fully functional, to promote specific web analytic tools in exchange for the monetary compensation from developers, as a fee for the marketing service.
– **Increase system efficiency:** by allowing a user to get content he/she is looking for, the WARES recommender system can lower time spent for search, and lower an amount of data to be exchanged, thus lowering the costs of time and rising user's efficiency in selection appropriate web analytics tool.

Potentially a recommender system can provide several sorts of recommendations, from a single item or a simple list of items to a sequence of items. It is exactly what WARES do, because not every time user will want to see tens of recommendations about web analytics tools, most time it is "top 5" or "top 10" lists. A user will need to know a lot of ongoing stuff about recommended analytics tool, and theoretically here WARES could come in handy by providing, right away, explanations about the features, which suggested web analytic tool possess. As was explained in the section 1.1.3 there are many metrics, theoretically new users could be unaware about what they does and how to use them.

Next question is how the recommendations will be integrated with other content navigation features. In most cases, users will be offered with means to browse content in addition to getting recommendations. I am considering the following methods that can greatly enhance the user's experience of WARES:

- **Separate or integrated recommendations**: meant recommendations for the user are shown in the separate window (tab) of the user's default internet browser, as a main feature of the application, and since an application itself implemented as a web–

application such recommendations may also appear right away in the recommender system's main window, integrated.

- **Optional or a mandatory:** usage of the recommender system is a mandatory part of the interaction model, because it is the main feature of the designed application, without the possibility to recommend the application has no sense of existence. Following this option the WARES should work in the "background", meaning that there should be no context pop–up windows like "please wait recommending process in progress". If a user started WARES application, than he/she should get something on the screen while recommendations being prepared, e.g. "tip of the day" about web analytics tools or kind of a "did you know" interesting facts about capabilities of the WARES.

It is important to define targets for the performance of the system along a number of criteria. Not only these criteria will allow to evaluate the system once it is built, but they are also a key for selection of the proper algorithms. Many criteria for a comprehensive reference can be used, see [Picault *et al.,* 2011] some key ones should be:

- **Transparency and "explainability":** in the WARES, I think it is highly desirable to have a certain level of transparency and "explainability" and it should be implemented as an indispensable part of the system, e.g. in the form of pop-up tips, while pointing the results with a mouse, containing explanations why this particular web analytic tool was placed (rated) higher in the list than another.
- **Risk taking**: the recommendations made for the user should be with a high probability of liking, no "risky" items can be recommended. For the WARES it is a controversy point, as it limits "novelty" of recommended items, but I think for the professionals, who value their time and money, and of course they do, will be preferable to get a "reliable" recommendation with "high probability of liking", no "new" random items should be recommended, only these web analytic tools, which are highly corresponding to user's needs.
- **Response speed/performance**: in some cases, users want to see the fast reaction from the application and sometimes reaction speed could be more important than the accuracy of the produced results. For the WARES it is not a critical point, the system just should work without crashes and freezes.

- **Reliability**: for the WARES, this is not a bottleneck, it simply must work without crashes and freezes, it is not the donor organ matching system for the hospital where it should have the highest level of reliability.
- **Robustness to attacks**: if the recommender system has a vital commercial role, e.g. eBay or Amazon, it may be a subject to the various hacker attacks from the competitors or just from some hooligans, in order to lower its performance and results. In the case of the WARES there is no such a threat, because at this stage it is not a commercial system and it is not designed as a client-server application, where the server could be the attack target.

3.2.1.2 – Understanding the influence of the application implementation

In addition to the features described in the section above, some aspects of the application implementation also will have a considerable influence on the way how the recommender system will be designed from the point of view of interaction with devices and OS platforms where it will be runt.

Single or multiple devices: is the same application designed for the multi-platform use? e.g. user could launch it from his/her cell phone or tablet, PC. For now I am considering implementing WARES only for PC platform, and only for Microsoft Windows, but as a part of future work and improvements considering to implement mobile access and porting into another OS platforms.

Single or multiple users: if the device being used by the several users the application should have the profiles management option, or if it is a client-server application, how to store profiles? This impacts on the architecture of the RS, rising requirements for the different identification methods, e.g. using cookies or login and passwords. For the WARES there is no problems mentioned above, it simply has one single user, and every time when application is launched it stores all the necessary data in its local folder, without any personal profiles, only basic settings which are the same for anybody who lunches the application.

Application infrastructure: here the two cases can be identified, whether the application is accessed through a browser as a client-server application or it runs locally on the user device.

- Browser–based client-server application: in this case most of the processing usually is done on the server side, whence the client will then receive only the results from the server. Not the case for the WARES.
- Browser–based local application: when the application runs on a local computer. On the current stage, the WARES recommender system is a relatively simple low-resource consuming application, there is no need in enormous processing power of the relevant devices, but it is quite sensitive to the network connectivity. The network connection should be permanent. The data needed for producing recommendations is getting from the ontology, which is stored locally and filled while the application is running.

Screen "real estate": simply determines how much of the screen space will be taken by the recommender application. In some cases, the screen-space might be limited by the application's user's interface design, but for the WARES it is not a problem, as it runs in a browser's window, thus the amount of the screen-space is determined by the user, who shrink-elongate the browser's window as he/she considers necessary. For example, if the WARES would be a fixed-size window application and if it has provided more recommendations, it is obvious that total amount of needed screen-space will rise and at some point recommendations may go beyond the application's window borders, but WARES is a browser-based application, user could always scroll up-down and left-right to see the all text.

3.2.2 – User model

In the modern world of commercial applications understanding needs of its users is a key to success of any recommender system. Insights of the user's needs should be modeled as early as possible because it may impact on the final efficiency and overall usability of the recommender system as a part of the application. In this section, will be considered how to characterize users by some chosen properties that may have an impact on the WARES design in the future. The recommender system may face some difficulties or complete unusability if

user's needs were incorrectly identified or interpreted, list of all user's properties, proposed by the Jerome Picault in his article [Picault et al., 2011] is shown in table 6.

Table 6: General user model components [Picault *et al.,* 2011]

Model features	Possible values
Demographics information	Is present, not present
Goal existence and nature	implicit, explicit
Level of expectation	high, medium, low
Possible change of expectation over time	yes, no
Limited capabilities of user's device	CPU, memory size, screen resolution, etc.
Importance of user's situation	high, medium, low
Social environment	alone, in the group of other people
Trust and privacy concerns	high, medium, low

If we want to go deeply while developing a recommender system the user, as en entity, might be studied very deeply based on the international standard [ISO–13407], but we will stick to those proposed in the article "How to get the recommender out of the lab" [Picault et al., 2011].

Table 7: User model for the WARES recommender system

Demographics information	There is no discriminatory demographic factors, anyone could use this system
Goal existence and nature	Explicit goals are expressed through user queries; End-users have no other goal than being informed about new web analytic tools, which are reflected through queries.
Level of expectation	Medium to low
Possible change of expectation over time	Yes: expectations of users should increase when they progressively discover the benefits of personalized search functions
Limited capabilities of user device	No: typically there is no limitations for PS platforms
Importance of user's situation	Low: the recommender system does not concern about user's situation.
Social environment	Alone: by nature, selection of something, in this case - web analytics tool, is an individual activity, conducted by the decision-making person.
Trust and privacy concerns	Not considered, because proposed recommender system does not store any personal information.

It is essential to understand who will be the end–users, what he/she expects when using WARES, what are the user's central contextual factors surrounding the use of the system? Only by clearly answering on each of these questions the fundamental requirements for the system design and choice of the technology will become clear, all properties for the WARES user model is shown in table 7.

3.2.2.1 – Importance of understanding who are the users

It is important to concentrate on the identification of user's characteristics because it has special utility in terms of recommender system design. By identifying user's characteristics, mean building a portrait of the different groups of users through demographic information, such as *age, gender*, *job area*, *nationalities*, and *spoken languages*. Knowing these characteristics allows to start building a relationship with users and get an appreciation of their needs. Understanding who are the future users will help to resolve two major concerns: *understanding user's key identifying characteristics* and *user's skill levels and their prior experience with similar systems*.

Creation of user group clusters allows building simple recommendations based on demographics, this feature is commonly used in targeted advertising campaigns and define stereotypes of this group of users. Stereotyping techniques allow definition of a set of certain defining characteristics for a group of users, which may help when a new user introduced to the system. Predefined stereotypes could be assigned for new users, based on their personal data, which allows activating a set of default preferences that may be further refined over time thanks to user's profile adaptation methods [Burke, 2002].

3.2.2.2 – Understanding user's motivation, goals and expectations

Knowing goals and motivations is very important for me as the designer of the recommender system. This knowledge may help to understand is an application could satisfy potential users or not. For example, the eBay is offering to its users the possibility get recommendations while browsing/buying item. From the user's point of view, while using this recommendations the goal is to buy or not to buy these items, while eBay's goal is to motivate users to buy more, by showing "relevant" items. We need to try to identify and to understand user's motivation and goals behind his/her actions, in order to make recommender system user-friendly and in the future to improve user's experience and the overall outcome of the system [Picault et al., 2011].

The goal: could be an implicit, when a user is "not sure" why he/she is using the recommender system, in this case, he/she may be offered a certain set of actions within a

recommender application, or the goal might be inferred during the interaction process of the recommender system with a user. Or it might be *explicit,* when user knows for sure what he/she will do within the recommender. For the WARES the goal is defined as an explicit because the users already know why they are using this recommender system (to get recommendations concerning the web analytics tools).

Level of expectation: the proposed recommender system is totally new, and at this point designed not for the commercial use, that is why the level of the expectations from the users might be "medium", means the recommender system returning "some good items" or "low", means the user does not expect outstanding results but expects the recommender system at least just working. User's expectations also may change over time, in the process of acquaintance with the system or with increases of their own needs.

3.2.2.3 – Understanding user's context

The last part to be taken into account are the various contextual issues surrounding the use of the recommender system [Picault *et al.,* 2011]:

User's device: the first consideration is "what device will be used by the user to access the recommender system"? For example, if this is a mobile device (smartphone or a tablet) we should consider using minimalistic user's interface. For the WARES recommender system, the device which will be used is the laptop or desktop computer, so there is basically no limitations on the design implementation.

Situation of interaction: situational considerations may include user's current location, where the user using the system – on the workplace in the formal setting or at home while being relaxed. Temporal factors, e.g. relevance of given recommendations during the certain period of time, etc. For the WARES recommender system, there is no limitation by situational interaction, because it recommends content which could be browsed anywhere and does not impose any limitations for its users.

Social environment: environment, where the use of the recommender system is usually carried out. Is it usually done alone or within some social group of other people, is there age

restrictions, etc. This may result in some design decision resulting in the final implementation, e.g. data collection methods or recommendations presentation methods. And again for the WARES there are no limitations on this factor, the content of its recommendations is not rated as “adult” or “disturbing”, so anybody could use the system anywhere, in the workplace in the office, with friends or even in the family circle. But we think it will be used mostly alone by the one person, because by nature, selection of something, in this case - web analytics tool, is an individual activity, conducted by the decision-making person.

3.2.3 – Data Model

The last thing, that should be studied, are the characteristics of the items with which the WARES recommender system will work. In table 8 is shown general data model components and all the characteristics of the data model, proposed by Jerome Picault in his article [Picault *et al.,* 2011] and to be considered in this the section. It is important to consider data model because it helps to identify the main characteristics of data that may influence the design of the recommender system.

Table 8: General data model components [Picault *et al.,* 2011]

Model feature	Possible values
Data type	structured, semi–structured, unstructured
Metadata quality and quantity	high, medium, low
Metadata expressiveness	keyword–based, semantic–based
Description based on standards	yes, no
Volume of items	tens, thousands, millions ,etc.
Diversity of items	homogeneous, heterogeneous
Distribution of items	“long–tail”, mainstream
Stability vs. persistence of items	stable, changing, changing a lot
User ratings	implicit, explicit, none

Not all components from the table above could be used in WARES, because of specificity connected with web analytics tools, and because of the need to extract these data from the unstructured textual documents (web pages). In table 9 are listed all data model components which are used in the WARES recommender system.

Table 9: Data model for WARES recommender system

Data type	Unstructured, data should be extracted from the web pages, describing particular web analytic tools.
Metadata quality and quantity	Low: poor expressiveness almost without any semantic annotation of content, many metadata should be extracted by the semi-automatic means.
Metadata expressiveness	Keyword-based: most of the information about web analytic tool gathered by filtering keywords and semantic text analysis in the body of web pages.
Description based on standards	Yes: proposed ontology for web analytics tools is used.
Volume of items	Medium: depending on how long recommender system is working, and how many web analytics tools were found during this period.
Diversity of items	Homogeneous content composed of web analytic tools.
Distribution of items	Mainstream formed of web analytics tools.
Stability vs. persistence of items	Stable: most of the web analytics tools, found during the web search process, remains unchanged in the "short" period of time, while recommendations are made.
User ratings	Explicit or none: user builds his/her own preference table during first interaction with recommender system when he/she asked about initial preferences.

3.2.3.1 – Understanding the type of available data to describe items

According to the Jerome Picault [Picault *et al.,* 2011], the main aim of the data model is to support the development of the recommender system by providing the insights about the data, which it uses. There exist many ways to describe data sources, but the best-known approach focuses on the three types of data: structured, semi-structured and unstructured data.

Unstructured data: in the case of the WARES recommender system, we are dealing with unstructured data, where an item can be represented only in unstructured form, it means that the data does not have a standardized data model e.g. in our case - unstructured text. In this case a certain preprocessing need to be done to extract significant keywords, or concepts, which are helping to distinguish relevant items from other textual information. For example, Apache Lucene [Lucene, 2016] allows extracting keywords from unstructured text.

The use of *unstructured data* has many impacts and limitations on the recommender system's design. First of all it reflects on the algorithms, which might be used, dealing with

unstructured data excludes a full set of recommender algorithm families, e.g. Bayesian models. To work with such date we first need to extract keywords, see section 3.3.2 for details, then build and item profile (ontology) and only after this manipulations we could start producing recommendations by manipulating received data in the way we do with the structured data, using of a vector of keywords representing user's profile.

3.2.3.2 – Understanding the quality/quantity of data

Quality and quantity of the data are important performance factors for any recommender system, especially when using a content–based recommender approach, like we do in the case of WARES, performance of the recommender system depends on data quality, precision of the recommendations depends on quantity of available data.

Quality: in general, an item data is considered "high quality" if it enables one item to be distinguished from another. The quality feature of the data is a major *"hot"* point of the design for the WARES recommender system, it depends on the web pages filtering, using a certain set of keywords. I need to balance the accuracy of the produced recommendations and the recommender system performance in terms of time to respond, storage capacity and processing costs. For example, if we prefers the "best" performance then we have to introduce some constraints on the architecture and avoid implementing the RS on a *lightweight client* (a device that depends heavily on another computer, its server, to fulfill its computational roles). Alternatively, I can choose to perform recommendations using a two–step algorithm, distributed between the server and the client device [Picault *et al.,* 2011].

Expressiveness: the data must reflect user's points of view on items, and represent what users consider as differentiating characteristics of items. The expressiveness of the data is crucial to the whole performance of the recommender system. The data, described using semantic concepts enables us to use more sophisticated recommender algorithms, such as *ontology–content–based filtering method* [Shoval *et al.,* 2008] that takes into account the existence of "related" items according to their position in the ontology hierarchy. Or spreading of semantic preferences, i.e. the extension of ontology–based user profiles through the semantic relations of the domain ontologies, as described in [Sieg *et al.,* 2007]. However, it exists certain technical

issues: semantic reasoning is increasing overall processing time and if the given data are not considered "sufficiently" semantically described it may also resulting in inaccurate recommendations.

Quantity: the amount of data is an important factor to consider: few data may lead to inaccurate recommendations, whereas too much metadata may lead to useless processing and overall slowdown. In addition, data description may vary in terms of degree of precision (*depth*), variety of description (*breadth*) [Picault *et al.,* 2011]. The risk with superficial description is to propose items to users that do not correspond exactly to what they expect. Very in–depth descriptions may reinforce some drawbacks of the content–based filtering algorithms, in particular in terms of overspecialization.

Description based on standard: regardless of their level of expressiveness, the data can be described in different ways using various standards such as: Dublin Core, MPEG, etc. For the WARES recommender system the date is described by the ontology, proposed in this paper.

3.2.3.3 – Understanding the properties of the item set

Volume of items: in addition to the quantity of the relevant data per item, we should consider the volume of items in the data set. It represents an important factor in the choice of a recommender system technique. The size of the data set is crucial for the collaborative-filtering techniques to compute correlations efficiently, while the content–based algorithms can cope with a smaller data set, this is another reason why for the WARES was decided to use content-based approach [Picault *et al.,* 2011].

Distribution of items: it is also essential to consider how items are distributed among the data set. In the WARES recommender system the items are distributed as a mainstream, meaning there is no high proportion of items annotated with some special concepts "action", all items are forming a set where each item is as important as everyone else. The level of depth of the annotations of the data is "in order" and all data is considered as of equal quality. [Picault *et al.,* 2011].

User ratings: existence of user's ratings and their amount influences on the type of techniques which could be selected for the recommender system, as in the WARES recommender system there are no initial ratings made by other users, the user may rate or may not rate all the recommended items after the first recommendation session, for the following filtering process, if desired. If there exists no user ratings related to the items inside a data set then this excludes the whole family of collaborative filtering methods.

3.2.4 – Summary on Recommender Environment and how it will be used further

The three models considered above, help us to understand the environment of the developed recommender system. For each model and for each feature/property, was proposed guidelines to define requirements and constraints for the WARES. To understand the importance of those features was used a two–step method:

- **Identify the dependencies between features**: by considering all the properties we were able to find out which are relevant and might be used, and which are not may be used in the WARES recommender system. This helped help us to understand how a change of the one feature may affect the overall recommender environment. For example, how changing the type of the application from client-server to local may change the recommender system's device platform.
- **Identify key features of the models**: those that have the most significant impact on the recommender system's architecture, affect the choice of algorithm, etc. For example, the type of data which will be used (unstructured data) determined the concept of further development for the WARES recommender system, thus it was confirmed, that using the content-based approach is preferable than using a collaborative-filtering family of methods.

In this section, while describing relevant features and properties also were identified the constraints, which should be taken into account, when designing the WARES recommender system.

3.3 – Architecture of the WARES recommender system

The purpose of the WARES was described in the introduction and relying on models considered in the first part of chapter 3, I can now start describing WARES itself. Three models: application model, user model, and data model for WARES was shown in the tables 5, 7 and 9 and the Architecture is shown in figure 17.

Worth emphasizing that in practice, these three models by themselves will not allow having a perfect design at the first go. However, they are a very useful support for an iterative design methodology, which is the best way to go beyond technical excellence and reach the goal that matters in the end: user satisfaction.

The architecture of the proposed WARES recommender system is shown in figure 17. The first block is the part of user interface, it will be described below in the section 3.3.1, it is responsible for gathering user's expectations and desirable parameters about his "ideal" web analytic tool, this parameters, along with basic search parameters, will be used during the online search. Second block is an online search itself, which is responsible for filling ontology with all necessary data for recommendation process; see section 3.3.2 for details. Block 3 is an ontology itself, which is used for the description of web analytic tools as a class and as a database, storing all data necessary for recommendation process, see section 3.3.3. Block 4 is responsible for making recommendations and uses an algorithm described in the section 3.3.4, this algorithm works with data collected through the steps in block 2 and 3.

In figure 17, block 3 is connected with block 2 and 4 with two–way arrows, which means that there is a constant information exchange between this three blocks. It is essential for recommender system functioning, because to produce recommendations block 4 needs access to the data, stored in the ontology and block 2 needs access to the ontology in order to fill it with new data. Block 5 is a part of the user interface, which presents recommendations to a user in the application window as a list of URLs with a short description of selected web analytic tools, with their key features respectively.

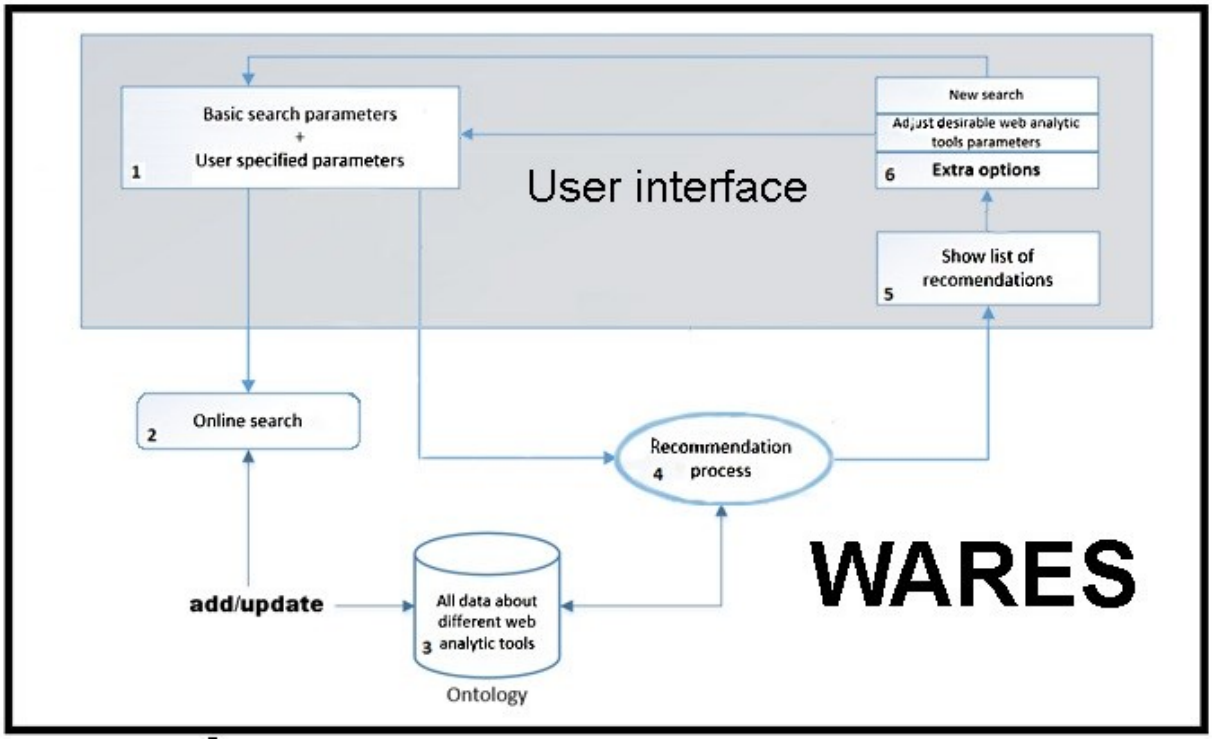


Figure 16: Architecture of the WARES recommender system for web analytic tools

Finally, the block 6 offers clarifications about received results, it means if user not satisfied with received recommendations he/she will be granted with possibility rate received results on the scale from 0 to 5 and then will have a choice to use previously received results with ratings for "refinement", along with possibility to start a new recommendation search.

3.3.1 – User's interface

On figure 17 the blocks 1, 5 and 6 are the part of user interface. Described in general user model in the section 3.3 the recommender system should be implemented as a web application, which means all input data and recommendations itself will be shown in the web browser's window.

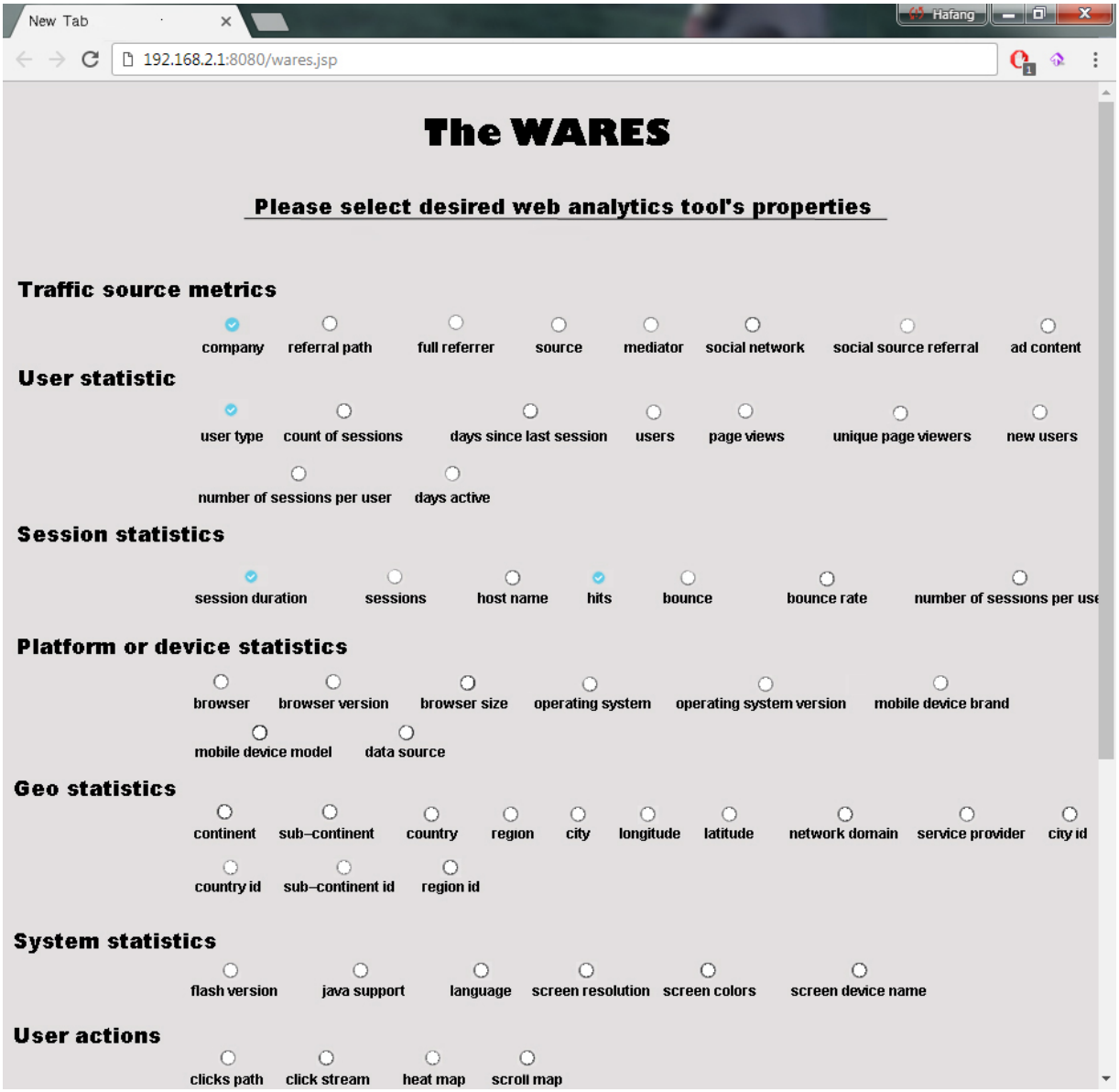


Figure 17: the WARES recommender settings 1

As a starting parameter for the online search, recommender system has a phrase “web analytics”, it is hidden inside the application code, if the user specified additional parameters, the system concatenates starting parameter string with these “user-specified parameters”. “User specified parameters” could contain all possible values described in detail in the section 3.3.3, sub-section “Define the properties of classes”. All possible values represented as a text with “checkbox” near it, see figure 18, if the user wants to include one or another property he/she should simply check it and the system will know that this property is “important”. Also a user

will be asked, how many results he/she would like to see after recommender process is completed, see figure 19.

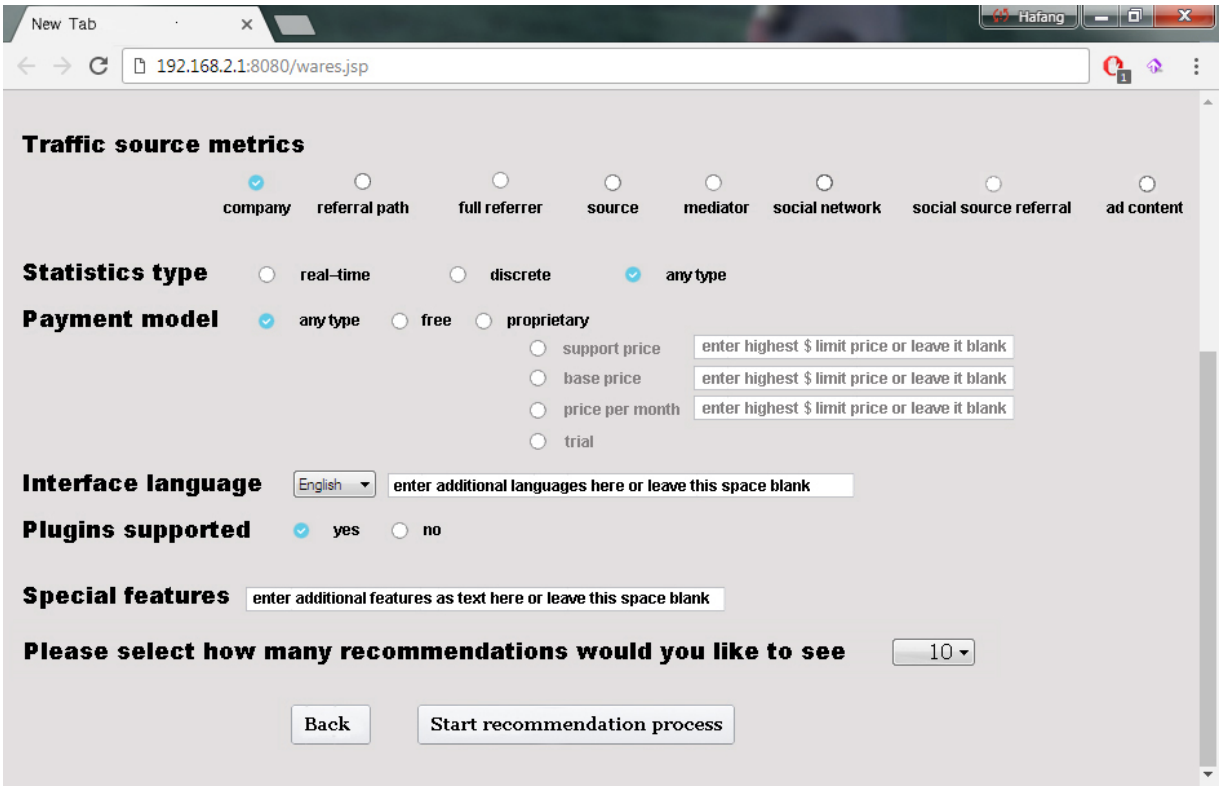


Figure 18: WARES recommendation settings 2

Next, during the filtering process, see figure 21, the system will "favor" pages containing in their body text with properties marked by the user. These "marked" pages will be listed as those, which represent certain web analytic tool and used during recommendation process, along with other information from the ontology. And finally user could enter anything he/she think important, about desirable web analytic tool, which he/she is looking for, e.g. a remark "free" or "reliable", this text will be added to the starting parameter, so the final string for search engine will be "web analytics, free" or "web analytics, reliable".

When the recommender process is complete, see block 5 in figure 17, the application will show a list of the links to the corresponding web analytic tools, which were recommended by the system, see figure 20, ordered accordingly to user's requirements, formulated in the block 1. It is meant, if a recommended web analytic tool possess all the properties selected by the user

in the section “user-specified parameters” then this link will be shown in the first place in the list of recommendations, and so on, all links are shown in the order sorted by the highest rating corresponding to “user-specified parameters”. If a recommended web analytic tool has fewer properties corresponding to the “user-specified parameters”, it will be displayed after all links which have all desirable properties, the fewer properties web analytic tool has, lower the place in the list it will get.

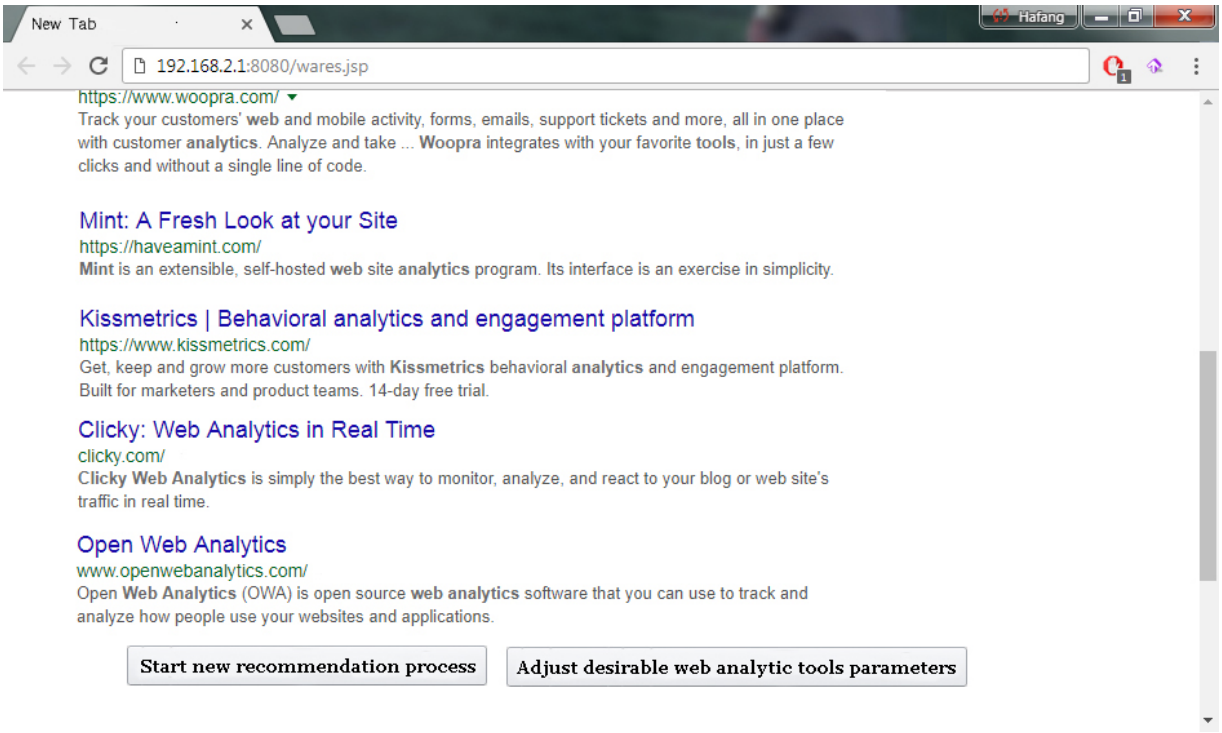


Figure 19: WARES recommendations results

By clicking on each of link, the user can see the main page of the selected web analytic tool, thus the user does not need any explanations from the recommender system, about why this particular web analytic tool was recommended, he will see all the needed information by himself on the developer’s website.

Finally the block 6 allows to select what to do next, simply asking the user, would he/she start a new search with new parameters or would like to modify/change previously entered properties to narrow (extend) received results of the previous search and recommendations. If user selected “adjust desirable web analytic tools parameters”, he/she will see web page with

previously checked “user-specified parameters”, and will be able to modify whatever he/she wants, then new parameters will be applied to previously obtained recommendations, except steps 1, 2 and 3, see figure 18. This process could be repeated as many times as needed until the user closes WARES application or chooses to start a new search.

3.3.2 – Online search and data mining in WARES

One of the key features of WARES recommender system that it uses an online search for making recommendations. Online – mean using web search engine for retrieving a list of web sites containing potentially useful data about web analytics tools, for the future recommending process. Theoretically, there is a possibility to create a new web–search engine, or use one of the existing frameworks of available open–source web search engines, but this task itself is so big and complex that it appears to be a topic for a separate thesis! That is why was considered to use already implemented excellent search engine like Google, so for this particular task was decided not to develop own search engine and took advantage of already well–established Google engine. The usage of Google search is pretty simple, e.g. using Java library jsoup [jsoup, 2016]:

```
Document doc = Jsoup
        .connect("https://www.google.com/search?q=Request&num=5");
        .userAgent("Mozilla/5.0")
        .timeout(5000).get();
```

Where ***Request*** is the word or set of words for input in Google search bar, and “*&num=5*” is the number of retrieved search results, in this case it is 5. The results then being parsed and filters out the domain names for each website, giving a list, e.g.:

https://piwik.org
https://www.woopra.com
https://metrica.yandex.com/
http://www.openwebanalytics.com/
http://www.google.com/analytics/

Next, comes the step 3 from figure 21, where for each website we need to build a site map, WARES recommender system will need it for extraction of data about web analytics tools. Usually, details about web analytic tool are not available on the main page, which in most cases just a "welcome" page, e.g. web analytic tool Piwik main page is https://piwik.org/ but key features about it could be found on the page https://piwik.org/features/. Other useful information, which is used in ontology, see section 3.3.3, could be hidden on the same site, but on the other pages, that is why we need to filter as many web pages as possible to find all possible useful data for "web analytic tool" class instances, during ontology building stage, see section 3.3.3. By retrieving a "site map", we will have access to all web pages of the current site, to do so we can use existing open–source code from *sitemapgen4j* Java library [sitemapgen4j, 2016], which can build XML site map for each site in format:

```
……
<url>
   <loc>https://piwik.org/features/</loc>
   <lastmod>2016–10–15T17:01+02:00</lastmod>
   <changefreq>never</changefreq>
   <priority>0.8</priority>
 </url>
……
```

Where each child element <*url*> containing information about web page, belonging to particular website, child element <*loc*> of <*url*> contains the actual URL link to one of this page. Having actual web pages URL's we can continue to the step 4, which is a text-filtering process of the entire web page body for the specific keywords describing web analytics tool, for the full list of keywords see section 3.3.3. If a keyword was found it is marked as "true" if, during the all parsing session for each web page of the website from the site map, corresponding to a certain web analytics tool, no entries were found it is marked as "false" for the currently analyzed web analytics tool. If at least 1 keyword were found, all the data is written into the ontology, as well all other found keywords characterizing the current web analytic tool instance, and search cycle is repeated again until all the results from the step 2 are being processed. Data retrieval process is shown in figure 21.

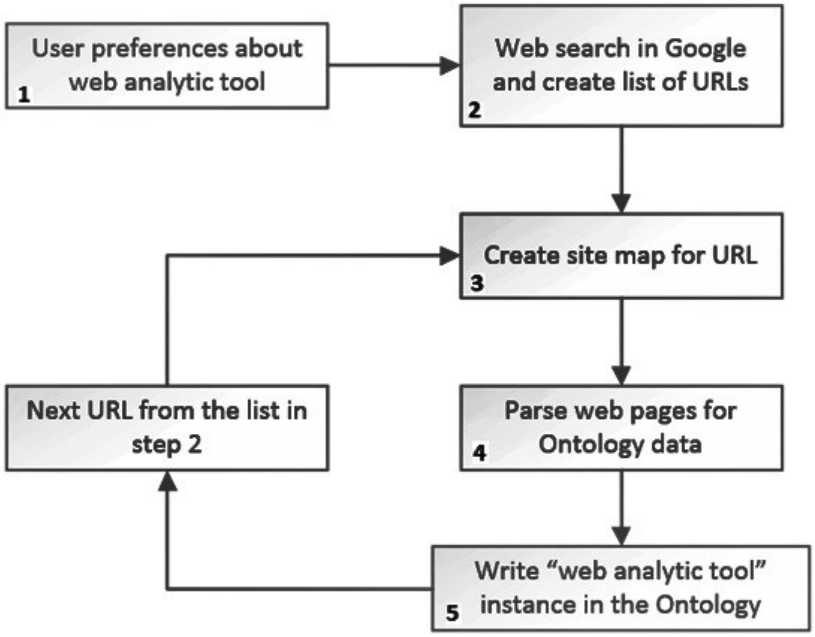


Figure 20: Search cycle "Filling the ontology"

Step 5 is actually a block of recommender system, which implements filtering by keywords from the ontology and some semantic analysis, because not all information could be expressed by keywords, e.g. data property from the ontology – "interface language", see next section for details. To be able to answer the question "which languages available for the user's interface?" text from the web page should be parsed using machine learning algorithms, e.g. Naïve Bayes is the simplest [McCallum, and Nigam, 1998].

3.3.3 – Proposed ontology for WARES

This ontology for my recommender system will have two purposes; first as classic ontology, it will describe the "web analytic tool" as a domain and second, it will be used as a database, storing data about web analytics tools, i.e. through class instances, for further analysis and making recommendations.

Ontologies are written in one of the OWL languages [OWL, 2012], designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than

that supported by XML [XML, 2008], RDF [RDF, 2014], and RDF Schema [RDFs, 2014] by providing additional vocabulary along with a formal semantics. OWL has been designed to meet this need for a Web Ontology Language. OWL is part of the growing stack of W3C recommendations related to the Semantic Web. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology.

Using the simple approach to creating web ontologies, which I learned from the course "Semantic Web", given in 2015 by the professor of the University of Montreal Ph. D. Guy Lapalme. I created a quite simple ontology for web analytics tools, using ontology editor Protégé 5 [Protégé, 2016]. The whole description of this ontology built in seven steps, with screenshots from Protégé 5 ontology editor, each step listed below describing an important piece of ontology structure.

Determine the domain and scope of the ontology, this section answers the following questions:

- What is the domain that the ontology will cover?
- For what we are going to use the ontology?
- For what types of questions the information in the ontology should provide answers?
- Who will use and maintain the ontology?

The answers to these questions may change during the ontology–design process, but at any given time, they help limit the scope of the model. Obviously, the domain is "web analytics tools". This ontology to be used for the WARES recommender system that suggests web analytic tools based on some user's preferences. The information in this ontology should provide an answer on the classification of existing web analytic tools. At this stage of the development, maintenance and support of this ontology is solely in my jurisdiction, but I do not exclude that it might be maintained by anyone who is interested in the idea of developing "web analytic tools" ontologies.

Consider reusing existing ontologies: it is almost always worth considering what someone else has done and checking if we can refine and extend existing sources for our particular domain and task. Reusing existing ontologies may be a requirement if the system

needs to interact with other applications that have already committed to particular ontologies or controlled vocabularies. Many ontologies are already available in electronic form and can be imported into an ontology–development Protégé. Before starting development of this ontology was considered using similar ontologies, but for such a domain as “web analytics tools” I was unable to find any, so the solution was to design my own ontology for this domain.

Enumerate important terms in the ontology: it is useful to write down a list of all terms we would like either to make statements about or to explain to a user. What are the terms we would like to talk about? What properties do those terms have? What would we like to say about those terms? Initially, it is important to get a comprehensive list of terms without worrying about the overlap between concepts they represent, relations among the terms, or any properties that the concepts may have. Actually, my ontology contains one main term "web analytic tool" which describes a class around which everything is built.

Define the classes and the class hierarchy: there are several possible approaches in developing a class hierarchy [Uschold and Gruninger, 1996]:

- A top–down development process starts with the definition of the most general concepts in the domain and subsequent specialization of the concepts. For example, let us consider general concepts of Wine as Food. Then we should specialize the Wine class by creating some of its subclasses: White wine, Red wine, Rosé wine. We can further categorize the Red wine class, for example, into Syrah, Red Burgundy, Cabernet Sauvignon, and so on.
- A bottom–up development process starts with the definition of the most specific classes, the leaves of the hierarchy, with a subsequent grouping of these classes into more general concepts. For example, we start by defining classes for Pauillac and Margaux wines. Then we could create a common superclass for these two classes Medoc, which in turn is a subclass of Bordeaux, etc.
- A combination development process is a combination of the top–down and bottom–up approaches. We define the more salient concepts first and then generalize and specialize them appropriately. We might start with a few top–level concepts such as Wine, and a few specific concepts, such as Margaux. We can then relate them to a middle–level concept, such as Medoc. Then we may want to generate all of the regional wine classes from France, thereby generating a number of middle–level concepts.

This particular ontology for the WARES recommender system, describing the “web analytic tool” as an item, does not have any specific classes, neither combinations, because here I have only one term “web analytic tool”, so technically the top–down approach is used, see figure 22.

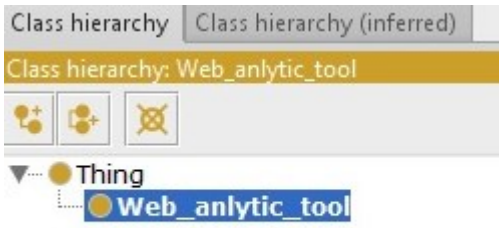


Figure 21: Class hierarchy for "web analytics tools" ontology

Define the properties of classes: class properties or slots, only one class in my case, alone will not provide enough information to answer the competency questions, so we need to describe the internal structure of the concept “web analytic tool”. As properties for “web analytic tool” will be used most of the properties described in the section 1.1.3, and some other, which was considered as “essential” for the domain “web analytics tools”. Here is a full list of properties, sorted by categories:

Metrics, which are includes:

a) “Traffic source metrics”: company, referral path, full referrer, source, mediator, social network, social source referral, ad content.
b) “User statistic”: user type, count of sessions, days since last session, users, page views, unique page viewers, new users, number of sessions per user, days active.
c) “Session statistics”: session duration, sessions, host name, hits, bounce, bounce rate, number of sessions per user.
d) “Platform or device statistics”: browser, browser version, browser size, operating system, operating system version, mobile device brand, mobile device model, data source.
e) “Geo statistics”: continent, sub–continent, country, region, city, longitude, latitude, network domain, service provider, city id, region id, country id, sub–continent id.

f) “System statistics”: flash version, java support, language, screen resolution, screen colors, screen device name.

g) “User actions”: clicks path, click stream, heat map, scroll map.

- “Statistics type”: real–time, discrete.
- “Payment model”: free, proprietary (trial, support price, base price, price per month)
- “Interface language”.
- “Plugins supported”.
- “Special features”.
- ID code.
- Name.

The overall properties structure is shown in figure 23, with some child sub–properties, e.g. “session statistics”, gathered into one parent sub–property, because the list is too long for one picture. Each property has its own data type and domain, the data type can vary from simple Boolean “true”, “false” which intends to show if current web analytic tool possesses this property or not, to more complex types as String, containing list of supported plugins or just a name of web analytics tool. Obviously only one domain – “web analytic tool”.

Define the facets of the slots: slots can have different facets describing the value type, allowed values, the number of the values (cardinality), and other features of the values the slot can take. For example, the value of an “ID code” slot is “1”. That is, “ID code” is a slot with value type integer. Some classes could have multiple instances, but in my case, there is only one class “web analytic tool”, that is why for simplicity was decided not to define any facets in object properties, but in the future, it is possible to extend this definition if ontology will continue to exist and develop.

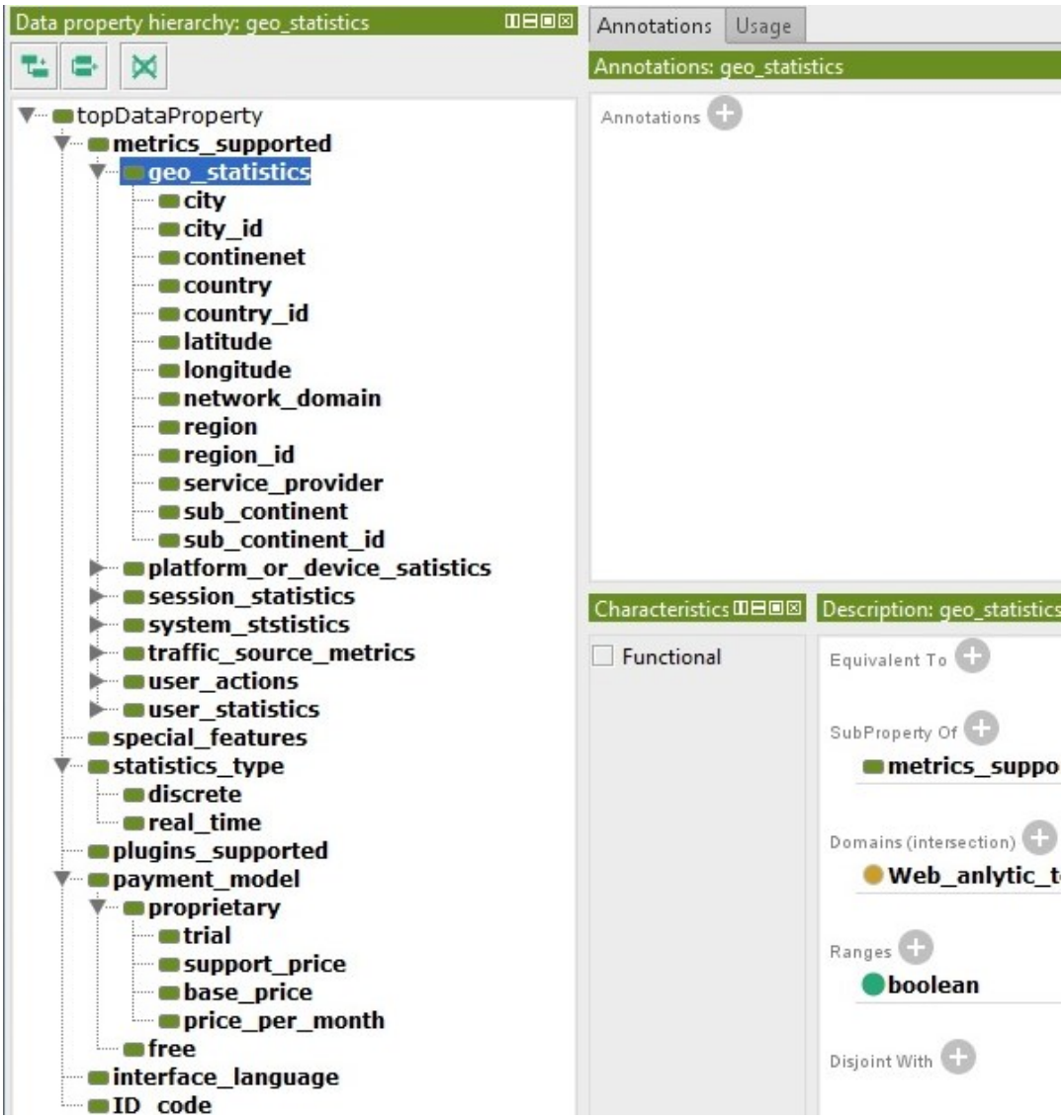


Figure 22: Properties defined for "web analytic tool" class

Create instances: the last step for each ontology is creating individual instances of classes in the hierarchy. Defining an individual instance of a class requires choosing a class, then creating an individual instance of that class and filling in the slot values for this instance. As an example, figure 24, we can see an instance of the class “web analytic tool” called “Woopra”, which in its turn represents the web analytics tool with the same name, considered in section 1.3 of this thesis.

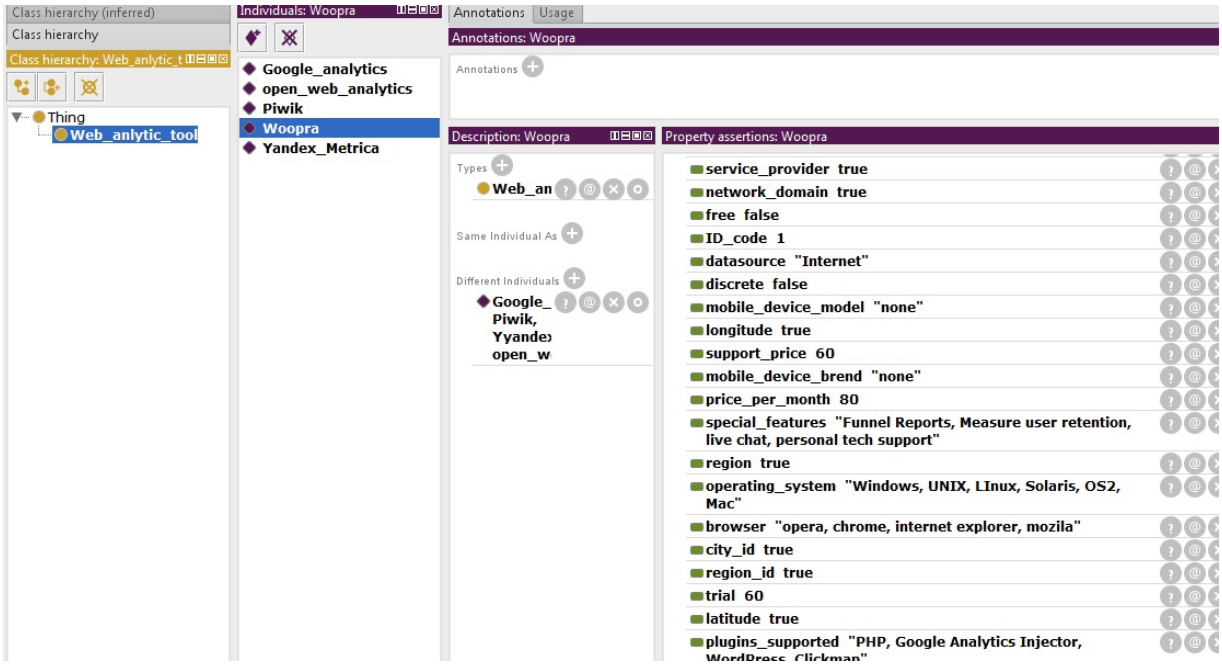


Figure 23: Example of an instance of the class "web analytic tool"

All the properties filled with appropriate values; you can see them on the right side of the figure but not all shown on the picture above, because as with the “data properties” the list is very long.

3.3.4 – Algorithm for the web analytic tools selection process.

Finally, the pseudo algorithm for making recommendations is shown below, with explanations for each step after it:

For a ∈ A

If (new search)

1: Generate weights w_i of each metric from ontology using ***TF–IDF***

2: Calculate ***n*** occurrences of the each term in ***a***

3: Calculate sum_{wi}

4: Calculate Sum_x

5: recommend web tools ordered by $max(max_a(n \text{ in } A) + \max(sum_{wi} + Sum_x))$

6: ask user rate results with r_i

Else (refinement)

7: ask for new parameters $param_i$ for refinement

8: do steps 2, 3 and 4 but with additional parameters $param_i$

9: calculate Pearson’s correlation p_{ij} for new results with $param_i$

10: recommend web tools ordered by $\max(p_{ij})$

End for

Here ***a*** is the web analytic tool instance with all metrics retrieved during the step 2, see figure 17, and written into ontology during step 3, see figure 17, described in the proposed ontology, see section 3.3.3 and ***A*** is the set of all ***a***.

- **Step 1:** Calculating word weights for each term, representing metrics describing web analytic tool. This is made during the stage 2, see fig. 17, because it saves time and memory resources and could be done while processing text from the web pages associated with each web analytic tool. We need this w_i because it will help us detect the significance of each metric for current web analytic tool, where ***i*** is the metric's index number.
- **Step 2:** Calculating a total number of terms specified by the user on the stage 1, see fig. 17, which was found during the web page filtering process for each web analytic tool a_i, this data will be used during the step 5.
- **Step 3:** Calculation sum_{wi} which is the sum of all metrics found during the stage 2, see fig. 17, on the website for current web analytic tool a_i.
- **Step 4:** Calculation sum_x number of inputs of additional parameters if they was specified by the user, for each web analytic tool a_i on a corresponding website.
- **Step 5:** Making recommendations and ordering them by $max(max_a(n \text{ in } A) + \max(sum_{wi} + Sum_x))$ which represents the web analytic tool with highest number of occurrences of terms w_i selected by user from the desirable metrics list selected by user on the stage 1, see figure 17, with additional parameters, if specified, on the website corresponding to a_i. This means the results will be ordered by decreasing, the lower the max score, lower the position or the web analytics tool in the final recommendations list.
- **Step 6:** Ask user to rate given recommendation scaling from 0 to 5, if he/she wants to make refinement of received results.
- **Step 7:** Ask additional parameters for refinement, it could be new keywords specified by user or new metrics of the desired web analytic tool from the list of all available metrics, note that user can add or remove metrics, anyway this new criterion will be saved as $param_i$.
- **Step 8:** Repeat steps 2 , 3 and 4 but with additional parameters $param_i$

- **Step 9:** Calculate Pearson's correlation p_{ij} for new results with $param_i$ and initially received recommendations rated by the user, after this step this results became "initial" if the user wants to continue "refinement".
- **Step 10:** Gives a user the new set of recommendations ordered by $\max(p_{ij})$

3.3.5 – Sample script explaining usage of WARES recommender system.

So let us imagine a scenario, where you are a new user, what he/she should do if he/she want to start using this recommender system but this is his/her first time, how to use it? This section explains how to use WARES recommender system.

First, the user should download the latest version of the WARES recommender system from developers website (it does not exist for this moment, to be done in the future, this is just an imaginary explanation). The recommender system is build using Java SE Runtime Environment 7, so the user must ensure that he/she has the latest Java machine installed on his/her computer. After installation process complete the user should launch an application, he/she will see his default Internet browser opened and in the new tab he/she will see the welcome screen of the WARES. After pressing the start button in the center the user will see one button offering to start a new recommendation search along with the help/instruction button and the third button offering the refinement of the previous results, which will be inactive and will activate only after the first recommendation search is finished. Pressing "new search" will open another view in the same tab offering to check desirable features what he/she wants to see in the web analytics tool(s). Selection of desirable features is very simple, user just need to click on the checkbox with desirable feature (metric), there are 69 most common web analytics tools features, which user can check, and at the end of the page there is a textbox in which user could enter some "special" feature which is not on the list, but he/she wants it to be present in web analytic tool. By pointing the mouse cursor on one of the features user will see a tooltip with a short description of the indicated feature, this may help him/her to decide what to check. If the user does not know what he/she want he/she can just leave all checkboxes unchecked and empty text boxes for additional keywords and number of received recommendations, the system will do the search with default parameters.

After he/she checked all desirable features, or leaved fields empty, he/she should enter a desirable number of results, which he/she will get after WARES accomplish the search, and press “next”. Then the user should wait for a while before the list of recommendation appears in the new tab, time of waiting depending on the number of desirable results and on the user's Internet connection bandwidth. Finally, the user will see a web page with recommendations structured like an ordinary Google search results, the “most relevant” web analytics tools will be on top of the list and less relevant will be lower on the list with a short description below the link. By clicking on the web analytic tool name from the list, which at the same time is the link, the user will be redirected to the main page (welcome page) of the web analytics tool’s website where he/she can download it or look for additional information. If the user is not satisfied with a given results he/she can click “refinement” button which became active after the first search or chose to start a new search with a new parameter.

If the user decided to “refine” existing results, after clicking on the “refine” button he/she will see the page with his previous choices and will be offered to check/uncheck additional features and enter/remove additional keywords from the textbox for additional features and will be asked to rate previous results for compliance with his/her expectations using scale from 0 to 5 and then press “next” again. The system will do the recommender search among previous results and present a new list of web analytic tools. The user could do refinement as many times as he/she wants or start a new search or simply close an application by clicking on the small red cross in the upper right corner.

3.4 – Conclusion

In in the beginning of this chapter were reviewed the set of “best practices” in creating recommender systems, showing a general structure of the content-based recommender system. Next through the section 3.2, were formulated requirements and constraints for the WARES recommender system, along with different ways of its implementation. Model-based approach was selected, using template design from the article of Jerome Picault [Picault et al., 2011], the WARES was described by the three models: data model, user model, and application model, for each of the models, were explained how they are used in the WARES, alongside with selected model’s components, transferring the properties of these models to the WARES recommender system. Finally, in section 3.3 an architecture of the WARES recommender system was presented. Through sections 3.3.1 – 3.3.5 was explained how each part of proposed architecture works.

Chapter 4 – Evaluation and validation of the WARES recommender system

Design and implementation is an important part of the development but evaluation, and testing also is an integral part. Based on a set of properties that are relevant to the WARES it will be a mostly theoretical evaluation because I do not have enough users and their ratings to perform a "full-scale" test, although some I did some real tests, giving to try this system to only 17 users.

The classic way of the evaluation of any recommender systems is to show its prediction power, which means to evaluate their ability to predict and to rate related content, it is important, but today some people [Gunawardana and Shani, 2011] believes it became a secondary requirement. What became more important is the user's experience with a recommender system, because one way or other users are the central part of the system, it is created for them, not only for some abstract results in order to get 99.9% of accuracy but first of all to get satisfaction. In many applications people use a recommendation system for more than an exact anticipation of their tastes, or for confirmation of what they already knew, but wanted to hear it from someone else just to be sure "I'm not alone who think so".

Often it is easiest to perform offline experiments using existing data sets and a protocol that models user behavior to estimate recommender performance measures such as prediction accuracy, but for the WARES recommender system, it is impossible to use existing datasets, because exists datasets for movies, for image recognition, voice samples, but there is no such dataset for the domain "web analytics", so there will be no test like this. A more expensive option is a user study, where a small set of users is asked to perform a set of tasks using the system, typically answering questions afterward about their experience, it was asked some of my friends to evaluate the system and try to rate it somehow, see section 4.2 for details. Finally, a large–scale experiments could be run on a fully deployed system, which called online experiments. Such experiments evaluate many aspects of recommender system's performance on real users who are oblivious to the conducted experiment, but it is also not an option because the proposed system never been deployed like a real e-commerce application.

4.1 – Validation of the WARES recommender system

In order to test the accuracy of the proposed algorithm, see section 3.3.4, we used a leave-one-out cross-validation with a specifically created list of the test users and their test ratings. We randomly select a user and a web analytic tool rated by that user, we assume that the user has not rated yet current web analytic tool and we attempt to predict his rating. Finally, we compare the predicted rating with the actual rating, in order to evaluate the accuracy of the predictions. To evaluate the accuracy itself was used Mean Absolute Error, deriving following equation:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i|$$

Where $\boldsymbol{n}$ is the number of predictions, $\boldsymbol{f_i}$ is a prediction I and $\boldsymbol{y_i}$ is the actual value, so the MAE represents the average difference between the prediction and the actual value.

The data set on which the tests were performed was built using a pseudo-random generator, because The WARES does not contain sufficient data in order to perform valid tests and draw conclusive results, similarly with user's profiles was also generated in a pseudo-random manner and was pretty sparse and could not be used properly to test the approach. The data set was built in 2 steps, first 30 users were created, and then for each user ratings was randomized, for the total number of 30 web analytic tools. In order to reduce the random effect of the ratings and to create a certain correlation between the users we augment the initial dataset of 30 users in a pseudo-random manner, as a result, and additional set of 15 users, whose ratings were based on the ratings of the "initial" users, considering $\boldsymbol{R_{a,i}}$ is a set of ratings of the initial user $\boldsymbol{a}$ for a web analytics tool i. Then the set of ratings for the new user $\boldsymbol{j}$ is $\boldsymbol{R_{a,i}}$ + random value from a vector {-1,0,1}, based on the initial 30 users 15 more was created from with ratings randomly chosen from vectors: {-1.0.1}, {-2,-1,0,1,2}, {0,1,2}, {1,2,3}, {-2,-1,0}, {-2,-1,0,1}, and each vector at least once chosen in each case. As a result total number of 330 users were created (30 initial + 300 additional) and each user rated 30 web analytic tools.

In order to test proposed algorithm a test set of 150, "user/web analytic tool" pairs were selected randomly, test ratings were predicted and compared to actual ratings using MAE, using the proposed algorithm from the 3.3.4 and SVD as an alternative, see figure 25.

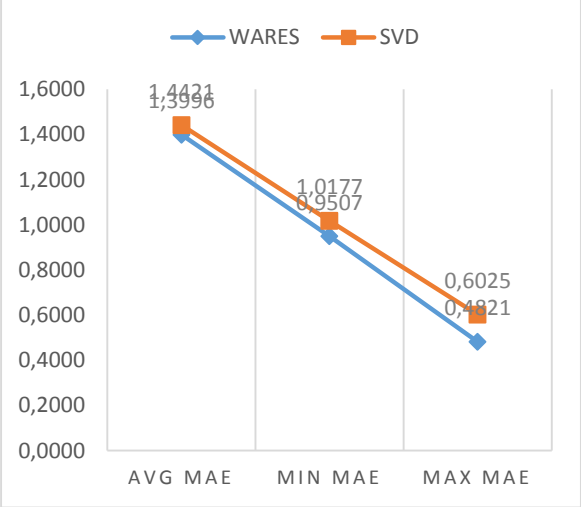


Figure 24: MAE comparison

In order to interpret the values of MAE, it is important to keep in mind the scale on which the ratings are performed, the MAE of 0.5 indicates that the predictions, on average, differed by the 0.5 of the actual rating. To evaluate the impact of this difference, keep in mind that difference in 0.5 on the scale from 1 to 5 is more significant than on the scale 1 to 20. As a result, we compared the variation of the MAE to the scale to assess its actual impact; that is MAE of 0.5 on the scale of 5 represents 10% when on the scale of 20 it only represents 2.5% and consequently lowers the impact on the accuracy. Figure 26 highlights the average value of MAE evaluated on the scale 5. Actually, the results are not bad at all but leaves some space for future improvements.

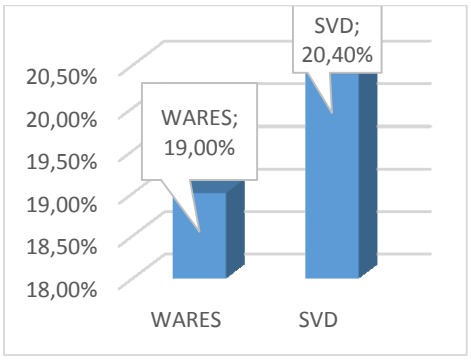


Figure 25: Average MAE interpretation

During the experiments was discovered that performance of the WARES mainly depends on the Internet bandwidth. Such a dependency is due to the specificity of the recommender system design, especially of the online search module, which regularly interchanges data with Google search engine. If the user decides to run WARES in the “quick” mode, which means that he/she wants to get recommendations immediately the system makes request to the search engine, receives fast response, but then when it accessing the web pages from the previously built “site map” there is a dramatic fall in productivity and further delays if the bandwidth is not enough. Figure 27 is showing the average time needed to process all 35 web pages for 50 site maps, using different bandwidth allocation.

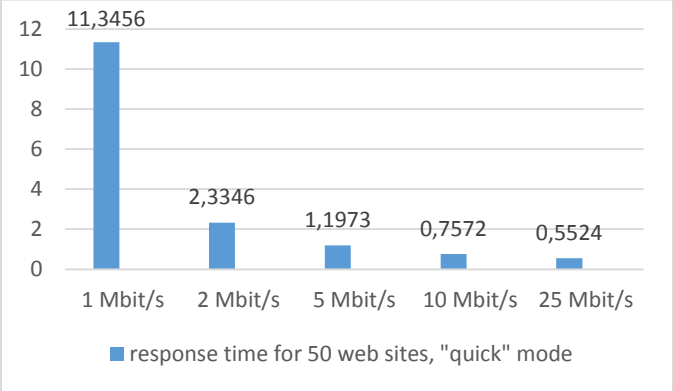


Figure 26: Time to complete recommendation process with different bandwidth allocation.

Another picture is observed if the system was started in the “background” mode, meaning the user set his/her preferences and left the system to work for a “long” time, an experiment: 30 minutes, user just got “instantly” without delays all the recommendations which he/she wanted to see, after the waiting time. During this experiment number of web sites was not limited to the number of recommendations was a lot more, and they were capped by 50 first relevant, this number could be changed via user interface, but academic research [Deursen et al., 2009] indicates that 91% of searchers do not go past page 1 of the search results and over 50% do not go past the first 3 results on page 1. On the other hand the precision of finding web analytic tools, corresponding to user's search criteria was higher (0.87%) for the background mode, comparing to the “quick” mode (0.68%) and probably the user satisfaction will be higher for the “background” mode, but we lack data due to a small number of users.

4.2 – Evaluation of the user's experience with WARES

We made a survey on a total number of 17 participants, see table 10, they are all my friends from the social network. The scenario as pretty simple: the WARES recommender system was given to each of them, alongside with questionnaire, and time of 3 days were given to complete the test. The main goal of the test was overall usability of the system and acceptance of the user's interface.

Table 10: Test participants information

Number of participants	17
Profiles of participants	17
Type of feedback	Explicit
Type of evaluation	Scales from 1 to 5
Female evaluators	6
Male evaluators	11
Number of recommendations per user	10 (170 total)

All users were asked to rate each aspect of the system on the scale from 1 to 5. The following questions were asked:

- **Overall rating**: how user evaluates proposed recommender system in general through his/her personal experience with it, during the test time.
- **Novelty**: degree of excitement, meant how surprising and how new was the topic of the proposed system, comparing to their personal experience with other recommender systems.
- **Convenience of the user's interface**: how easy was to understand how to use the system and, was it hard to navigate through the recommendation, how useful were the tooltips, etc.
- **Ease of use and intelligibility**: clarity of the content, is there was an ambiguity, or everything was clear through language, familiarity, aesthetics, forgiveness, efficiency, etc.
- **Stability**: means how often system crashed, is there were disconnections during the online search sessions, did the system hang?

- **Meeting expectations:** by user's personal feelings, does the system gives really relevant recommendations or not?

Table 11: How users rated the system

Overall rating	3.588
Novelty	4.751
Convenience of the user's interface	3.342
Ease of use and intelligibility	4.341
Stability	5
Meeting expectations	4.029

An interesting detail was discovered during the user's test: all users reported that the system consumes almost all the Internet bandwidth, meant that when an online search is in progress, the speed of downloading/uploading of any other content is dramatically lowered, meaning that the WARES tries to use all available bandwidth for its needs. Another interesting discovery was mistakes consisting of referencing irrelevant recommended contend due to giving URL's corresponding to the different blogs, describing or reviewing web analytic tools in details, while not actually appears to be a web analytics tool's home page it was helpful as a review, according to the testers quiz.

4.3 – Comparison of the WARES with other well–known systems on the market

Through the chapter 4, based on the "best practices", was explained how WARES recommender system could be evaluated and tested, of course, a more demonstrative way of evaluation – comparing things related to each other by some shared features. In case of recommender systems, theoretically, we could evaluate effectiveness, usability and many other parameters of the WARES recommender system, comparing it to already existing recommender systems. A good question arises – how can we compare recommender systems? No doubt, that in any field of science reproducibility is an important thing helping in conducting tests and comparison. But, in the field of recommender systems reproducibility almost unattainable because of specificity of the content, even it the same areas, such as recommending of scientific

articles, different recommender systems give different sets of recommendations. Several content–based filtering (CBF) and collaborative filtering (CF) approaches for research papers recommendations were tested [Beel and Breitinger, 2016]. In one experiment, CF and CBF performed similarly well, other experiments, CBF outperformed CF and in some more experiments, CF outperformed CBF.

Various authors showed that offline and online evaluations often provide contradictory results, and several more papers about various aspects of recommender systems evaluation have been published [Beel, 2015], [Beel and Langer, 2015]. Also was conclude that recommender systems research community is facing a crisis where a significant number of papers present results that contribute little to collective knowledge, often because the research lacks the evaluation to be properly judged and, hence, to provide meaningful contributions [Konstan and Adomavicius, 2013]. Why these contradictions occurred, even among very similar literature recommendation scenarios, remains widely unknown. The authors of the studies only mention a few potential reasons for the variations, such as different datasets or variations in the recommendation approaches. The current difficulties in reproducing recommender system results lead to a problematic situation for comparison. Developers who need an effective recommendation approach, and researchers who need a baseline against which to compare a novel approach, find little guidance in existing publications. So how WARES could be compared to other ones, e.g. well–known Amazon, eBay, Netflix, Pandora or YouTube? First of all to evaluate performance we need to use the same data set on all previously listed well–known recommender systems, but this is *impossible* because each of this systems specifically tailored for their narrow tasks, like WARES tailored for web analytic tools. As was highlighted above, it is really hard to compare not only my recommender system to another systems, but any recommender system in one domain to another recommender system in other domain. The few thing what we can do are: describe previously mentioned recommender systems (Amazon, eBay, Netflix, Pandora and YouTube) by algorithms what they use, number of users, recommender type (content–based, collaborative or hybrid), user's interface features, generated revenue, field of given recommendations, and some unique special features inherent to this systems, but cannot compare effectiveness by any numbers indicating accuracy, like MAE or RMSE.

Summarize this section, in table 12, you can see the comparative characteristics of considered well–known recommender system to one, proposed in this thesis.

Table 12: Comparative summary of well–known recommender systems and WARES

RS name	**RS type**	**number of users**	**field of recommendations**	**user interface**	**special features**	**revenue**
Amazon	CB item–to–item	~305m accounts	goods and services	web interface	Preview: "Search Inside the Book" before buying	135.98 billion
eBay	CB graph based	~167m accounts	goods and services	web interface	PayPal billing system	9 billion
Netflix	Hybrid, but mostly CBF	~95m accounts	Video streaming	Web interface	Possibility to see what your friends are watching now	8.8 billion
Pandora	Hybrid CB–CBF	~80m active users, ~250m registered	Music streaming	Web interface or special Application for PC or Mobile device	Possibility to see lyrics while listening to a song	1.2 billion
YouTube	Hybrid CB–CBF	~1.3b accounts	Video streaming	Web interface	Subtitles generation, video stabilization	9 billion
WARES	CBF	1	web analytics tools	web interface (browser based)	ontology for web analytic tools	n/a

4.4 – Conclusion

Because of the difficulty of the posed task of creating the recommender system, almost no real data from WARES were given, mostly theoretical guidelines on how to evaluate each aspect of the WARES recommender system. Was described the concerns that need to be addressed when designing offline and online experiments and user studies, outlined a few important measurements that one must take into account when designing experiments for the recommender system. In section 4.3 was explained why it is hard to compare one recommender system, e.g. WARES, to another recommender systems, but still manage to compare WARES recommender systems to the major "behemoths" on the market, resulting a table with a few common characteristics by which they could be compared.

Chapter 5 – Conclusion and future works

Proposed recommender system, WARES, is more likely to be a prototype, a concept. During my research, was considered hundreds of scientific articles about recommenders systems, but this is just a top of an iceberg. To implement proposed recommender system was essential to use various techniques, and recommendation process only one of them. Difficulties come from the ambitious goal defined from the beginning, a search of the textual information by keywords is itself a big area of information retrieval, search on the web just an extension of the semantic analysis.

Was considered two basic approaches how to make recommendations, the content–based and collaborative filtering, most notable methods and how they could be used in the proposed recommender system was explained in chapter 2. Here another problem arose, from the lack of real data, because all this method is centered on user's ratings, and in the WARES recommender system there is no users, the "cold start" problem. Partly this could be solved by saturating the system with "false" rating data, artificially generated by some pseudorandom number generator. But, in my case this is not a solution because these ratings should correspond to the real–existing websites, representing web analytics tools. This problem remains open and could serve as one of the directions for future works.

In chapter 3, was presented an architecture developed for the WARES recommender system, implementing well–known approach for application development based on three models: user model, data model and application model, each of which describes an important part of any application during the design process. In this case, this approach was applied to design a recommender system, but with minor changes, including the use of proposed ontology. A good contribution from this work could be an ontology, developed for the domain of web analytics tools, of course, this is not a new trend, ontologies started being used in applications long ago, but in case of WARES, was proposed an ontology for entirely new domain – web analytic tools. This could be a considerable innovation because during literature study I was unable to find any mentions about any existing ontology for web analytics. As a future work in this area, it is possible to focus on a more detailed elaboration of this ontology, not a secret that in last years, with an era of the Semantic Web ontologies become a common thing all around

the Web, and what is more important they could be reused and supplemented by anyone concerned about it. The ontologies on the Web ranges from large taxonomies categorizing websites, such as Google and Yahoo, to categorizations of products for sale and their features, such as on Amazon. Ontologies enable reuse of the domain knowledge, and by developing an ontology for the web analytics I could contribute to the WWW community or some other scientists who are working in the field of the WWW.

In the section 3.3.2 was described how online search for web analytics tools will work applied to the WARES recommender system. There was not enough time and resources for developing completely new specialized web–search engine for web analytics tools so was decided to use Google as a basis. But before that, there were attempts to implement something different, something new, based on the open–source web crawler Apache Lucene [Lucene, 2016], which is a full–featured text search engine library written entirely in Java programing language. It is a technology suitable for nearly any application, which requires full–text search, especially cross–platform. But it requires time for deployment, not only for understanding how it works and writing own code with modifications, it is least what was needed, but real "time" and resources. Means processing time, hard disk space, memory and CPU computation power, as a part of "crawling" process, the system should work days even months for indexation of existing resources on the Web. Only then, these indexes could be used for web analytics tools search and recommendations, as my own data. Which in its turn making recommender system structure more complex, by forcing usage of the client–server approach, and to some extent, it became similar to a "heavy" systems like Amazon. This could be a very good direction for the future work if funding will be found, or maybe someone else will decide to use the framework proposed in this paper for the full–scale commercial recommender system.

In chapter 4 was given a qualitative validation of the proposed recommender system, although during the testing process it was discovered some flaws in performance, which are connected with an Internet bandwidth. The system was tested on local computers of different users with different Internet speed connection, it was discovered that an optimal performance achieved with 5 megabit per second and higher. The accuracy of given recommendations is relatively high, it was rated 4 out of 5 stars during the test process with 17 participants, and most tested users found them helpful.

References

[Agrawal *et al.,* 1993] – R. Agrawal, T. Imielinski, and A. Swami – "Mining association rules between sets of items in large databases". SIGMOD Rec., 22(2), pp207–216, 1993.

[Agresti, 2010] – Agresti, A. "Analysis of Ordinal Categorical Data" (Second ed.). New York: John Wiley & Sons. 2010

[Amatriain *et al.,* 2009] – Amatriain, X., Pujol, J. M., & Oliver, N. "I like it... I like it not: Evaluating user ratings noise in recommender systems." *In International Conference on User Modeling, Adaptation, and Personalization,* pp. 247–258. Springer Berlin Heidelberg. 2009

[Anand and Mobasher, 2007] – Anand, S.S., Mobasher, B. "Contextual recommendation. In: From Web to Social Web: Discovering and Deploying User and Content Profiles", Lecture Notes in Computer Science, pp.142–160. Springer–Verlag (2007)

[Anderson, 2006] – Anderson, Chris. "The long tail: Why the future of business is selling less of more." Hachette Books, 2006.

[Baeza–Yates and Ribeiro–Neto, 1999] – Baeza–Yates, R., Ribeiro–Neto, B.: "Modern Information Retrieval". Addison–Wesley (1999)

[Beel and Breitinger, 2016] – Beel, Joeran, Corinna Breitinger, Stefan Langer, Andreas Lommatzsch, and Bela Gipp. "Towards reproducibility in recommender–systems research." User modeling and user–adapted interaction 26, no. 1, pp. 69–101. (2016)

[Beel and Langer, 2015] – Beel J., Langer S. – "A comparison of offline evaluations, online evaluations, and user studies in the context of research–paper recommender systems." *In Proceedings of the 19th International Conference on Theory and Practice of Digital Libraries (TPDL)*. Lecture Notes in Computer Science, pp. 153–168 (2015).

[Beel, 2015] – Beel Joeran – "Towards effective research–paper recommender systems and user modeling based on mind maps" PhD Thesis. Otto–von–Guericke University, Magdeburg (2015)

[Bell and Koren, 2007] – Bell, R., and Koren, Y., "Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights*", IEEE International Conference on Data Mining (ICDM'07)*, pp. 43–52, 2007.

[Bell *et al.,* 2007] – Bell, Robert, Yehuda Koren, and Chris Volinsky. "Modeling relationships at multiple scales to improve accuracy of large recommender systems." *In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 95–104. ACM, 2007.

[Bias and Mayhew, 1994] – Bias, R., Mayhew, D. "Cost–Justifying usability." Morgan Kaufman Publishing, 1994

[Billsus *et al.,* 2002] – Billsus, D., Brunk, C.A., Evans, C., Gladish, B., Pazzani, M. "Adaptive interfaces for ubiquitous web access." Communications of the ACM 45(5), pp34–38 (2002)

[Blanco–Fernandez *et al.,* 2008] – Blanco–Fernandez, Y., Pazos–Arias J. J., G.S.A., Ramos–Cabrer, M., Lopez–Nores, M. "Providing Entertainment by Content–based Filtering and Semantic Reasoning in Intelligent Recommender Systems." IEEE Transactions on Consumer Electronics 54(2), pp. 727–735, 2008

[Breese *et al.,* 1998] – Breese, J.S., Heckerman, D., Kadie, C. "Empirical analysis of predictive algorithms for collaborative filtering." *In: Proc. of the 14th Annual Conf. on Uncertainty in Artificial Intelligence*, pp. 43–52. Morgan Kaufmann (1998)

[Breese *et al.,* 1998] – Breese, J.S., Heckerman, D., Kadie, C.M. "Empirical analysis of predictive algorithms for collaborative filtering." In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 43–52 (1998)

[Burke, 2002] – Burke, R. "Hybrid recommender systems: Survey and experiments." User Modeling and User Adapted Interaction 12(4), pp. 331–370 (2002)

[Clifton, 2010] – Brian Clifton – "Advanced Web Metrics with Google Analytics, 2nd edition", Sybex, 2010

[Crestani and Lee, 2000] – Crestani F., Lee P.L. "Searching the Web by constrained spreading activation." Information Processing and Management, 36(4), pp. 585–605 (2000

[Davidson *et al.,* 2010] – James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston – "The YouTube video recommendation system", *in Proceedings of the fourth ACM conference on Recommender systems*, pp. 293–296, Barcelona, Spain — September 26 – 30, 2010

[Deerwester *et al.,* 1990] – Deerwester, S., Dumais, S., Furnas, G.W., Landauer, T.K. and Harshman, R., "Indexing by Latent Semantic Analysis", Journal of the Society for Information Science 41, pp. 391–407, 1990

[Deshpande and Karypis, 2004] – Deshpande M., Karypis G. "Item–based top–N recommendation algorithms". ACM Transaction on Information Systems 22(1), pp. 143–177, 2004

[Deursen *et al.,* 2009] – Van Deursen, Alexander J.A.M., and Jan A.G.M. Van Dijk. "Using the Internet: Skill related problems in users' online behavior." Interacting with computers #21.5, pp 393-402. 2009

[Domingos and Pizzani, 1997] – Domingos, P., Pazzani, M.J. "On the Optimality of the Simple Bayesian Classifier under Zero–One Loss." Machine Learning 29(2–3), pp. 103–130, 1997

[Falk, 2015] – Kim Falk, "Practical recommender system", manning publications, 2015

[Fischer, 2001] – Fischer, G. "User modeling in human–computer interaction. User Model." User–Adapted Interaction 11(1–2), pp. 65–86 (2001)

[Fredricks and Nelsen, 2007] – Fredricks, G.A., Nelsen, R.B. "On the relationship between spearman's rho and Kendall's tau for pairs of continuous random variables." Journal of Statistical Planning and Inference 137(7), pp 2143–2150 (2007)

[Gabrilovich and Markovitch, 2007] – Gabrilovich, E., Markovitch, S. "Computing Semantic Relatedness Using Wikipedia–based Explicit Semantic Analysis." *In: M.M. Veloso (ed.) Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 1606–1611 (2007)

[Gawesh *et al.,* 2010] – Gawesh J., Szomszor M., Kostkova P. "Comparison of implicit and explicit feedback from an online music recommendation service", *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, pp. 47–51 , Barcelona, Spain – 26 September 2010

[Gelman *et al.,* 1995] – Gelman, A., Carlin, J.B., Stern, H.S., and Rubin, D.B., "Bayesian Data Analysis." Chapman and Hall, 1995.

[Gomez–Uribe and N. Hunt, 2015] Gomez–Uribe, Carlos A., and Neil Hunt. "The Netflix recommender system: Algorithms, business value, and innovation." ACM Transactions on Management Information Systems (TMIS) 6, no. 4, p. 13 (2016)

[Gunawardana and Shani, 2011] – Shani, Guy, and Asela Gunawardana. "Evaluating recommendation systems." Recommender systems handbook, pp. 257–297. Springer US, 2011.

[Herlocker *et al.,* 1999] – Herlocker, J.L., Konstan, J.A., Borchers, A., and Riedl, J., "An Algorithmic Framework for Performing Collaborative Filtering", *Proc. 22nd ACM SIGIR Conference on Information Retrieval*, pp. 230–237, 1999.

[Herlocker *et al.,* 2004] – Herlocker, L., Konstan, J.A., Terveen, L.G., Riedl, J.T. "Evaluating Collaborative Filtering Recommender Systems." ACM Transactions on Information Systems 22(1), pp. 5–53, 2004

[Hu and Cercone, 2004] – Hu, X., Cercone, N. "A Data Warehouse/Online Analytic Processing Framework for Web Usage Mining and Business Intelligence Reporting" International Journal of Intelligent Systems, 19(7), pp. 585–606, 2004

[Huang *et al.,* 2016] – Huang Tony Cheng–Kui, Yen–Liang Chen, and Min–Chun Chen. "A novel recommendation model with Google similarity." Decision Support Systems 89, pp. 17–27. (2016)

[ISO–13407] – International Organization for Standardization (ISO): ISO 13407: "Human centered design processes for interactive systems."

[Jarvelin and Kekalainen, 2002] – Jarvelin, K., Kekalainen, J. "Cumulated gain–based evaluation of information retrieval techniques." ACM Transactions on Information Systems, 20(4), pp. 422–446 (2002).

[Kaushik, 2009] – Kaushik, A. "Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity" 1st ed. Indianapolis, John Wiley & Sons 2009

[Kendall and Gibbons, 1990] – Kendall, M., Gibbons, J.D.: "Rank Correlation Methods", 5th edition. Charles Griffin, 1990

[Kim and Yum, 2005] – Kim, D., and Yum, B., "Collaborative Filtering Based on Iterative Principal Component Analysis", Expert Systems with Applications 28, pp. 823–830. 2005

[Konstan and Adomavicius, 2013] – Konstan, J.A., Adomavicius, G. "Toward identification and adoption of best practices in algorithmic recommender systems research." *In: Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, pp. 23–28. ACM, New York, 2013

[Koren and Bell, 2011] – Koren Y., Bell R., "Advances in Collaborative Filtering" – Recommender Systems Handbook, pp. 145–186, springer 2011

[Koren, 2008] – Koren Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." *In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 426–434. ACM, 2008.

[Koren, 2010] – Koren, Y. "Collaborative filtering with temporal dynamics." Communications of the ACM, 53(4), pp. 89–97. 2010

[Lees–Miller *et al.,* 2008] – Lees–Miller, J., Anderson, F., Hoehn, B., Greiner, R. "Does Wikipedia Information Help Netflix Predictions?" *In: Seventh International Conference on Machine Learning and Applications (ICMLA)*, pp. 337–343. IEEE Computer Society (2008). ISBN 978–0–7695–3495–4

[Linden *et al.,* 2003] – Linden, G., Smith, B., and York, J., "Amazon.com Recommendations: Item–to–Item Collaborative Filtering", IEEE Internet Computing 7, pp. 76–80, 2003

[Lops *et al.,* 2011] – Pasquale Lops, Marco de Gemmis and Giovanni Semeraro, "Content–based Recommender Systems: State of the Art and Trend", Recommender systems handbook, Springer, 2011, pp. 73–106

[Lorrentz, 2016] – Pierre Lorrentz, "Artificial Neural Systems – Principle and Practice." Bentham Science Publishers, 2016

[Marlin et all, 2007] – Marlin, B.M., Zemel, R.S., Roweis, S., and Slaney, M., "Collaborative Filtering and the Missing at Random Assumption", *Proc. 23rd Conference on Uncertainty in Artificial Intelligence*, 2007

[McCallum, and Nigam, 1998] – McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naïve Bayes text classification." *AAAI–98 workshop on learning for text categorization.* Vol. 752. pp. 41–48, 1998.

[Mooney and Roy, 2000] – Mooney, R.J., Roy, L. "Content–Based Book Recommending Using Learning for Text Categorization." *In: Proceedings of the 5th ACM Conference on Digital Libraries*, pp. 195–204. ACM Press, New York, US, San Antonio, US (2000)

[Netflix, 2015] – Carlos A. Gomez–Uribe and Neil Hunt – "The Netflix Recommender System: Algorithms, Business Value, and Innovation", ACM Transactions on Management Information Systems, Vol. 6, No. 4, pp. 13, December 2015

[Oard and Kim, 1998] – Oard, D.W. and Kim, J., "Implicit Feedback for Recommender Systems", Proc. 5th DELOS Workshop on Filtering and Collaborative Filtering, pp. 31–36, 1998

[Picault and Ribiere, 2008] – Picault, J., Ribiere, M. "An empirical user profile adaptation mechanism that reacts to shifts of interests." *Submitted to the 18th European Conference on Artificial Intelligence* (2008).

[Picault *et al.,* 2011] – J. Picault, M. Ribiere, D. Bonnefoy, K. Mercer "How to get the recommender out of the lab", Recommender Systems handbook p 333–365, Springer 2011

[Pizzani and Billsus, 2007] – Pazzani, M.J., Billsus, D. "Content–Based Recommendation Systems." The Adaptive Web, Lecture Notes in Computer Science, vol. 4321, pp. 325–341, 2007

[Poirier *et al.,* 2010] – POIRIER D., FESSANT F., TELLIER I., "De la Classification d'Opinion à la Recommandation: l'Apport des Textes Communautaires", TAL : traitement automatique des langues, revue semestrielle de l'ATALA, vol. 51, no. 3, pp. 19–46, 2010.

[Salton, 1989] – Salton, G. "Automatic Text Processing". Addison–Wesley (1989)

[Sarwar *et al.,* 2001] – Sarwar, B., Karypis, G., Konstan, J., and Riedl, J., "Item–based Collaborative Filtering Recommendation Algorithms", *Proc. 10th International Conference on the World Wide Web*, pp. 285–295, 2001.

[Schein *et al.,* 2002] – Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M. "Methods and metrics for cold–start recommendations." *In: SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 253–260. ACM, New York, NY, USA (2002)

[Schwab *et al.,* 2001] – Schwab, Ingo, Alfred Kobsa, and Ivan Koychev. "Learning user interests through positive examples using content analysis and collaborative filtering." Internal Memo, GMD, St. Augustin, Germany (2001).

[Sebastiani, 2002] – Sebastiani, F, "Machine learning in automated text categorization", ACM computing surveys (CSUR), 34(1), pp.1–47, 2002

[Shani *et al.,* 2005] – Shani, G., Heckerman, D., Brafman, R.I. "An mdp–based recommender system." Journal of Machine Learning Research 6, pp. 1265–1295 (2005)

[Shardanand and Maes, 1995] – Shardanand, U., Maes, P. "Social information filtering: Algorithms for automating "word of mouth"." In: CHI '95: *Proc. of the SIGCHI Conf. on Human factors in Computing Systems*, pp. 210–217. ACM Press/Addison–Wesley Publishing Co., New York, NY, USA (1995)

[Sharma *et al.,* 2016] – Sharma, A., Jiang, J., Bommannavar, P., Larson, B., & Lin, J. "GraphJet: real–time content recommendations at twitter." *Proceedings of the VLDB Endowment,* 9(13), pp. 1281–1292. 2016

[Shoval *et al.,* 2008] – Shoval, P., Maidel, V., Shapira, B. "An ontology–content–based filtering method." International Journal of Information Theories and Applications (15), pp. 303–318, 2008

[Sieg *et al.,* 2007] – Sieg, A., Mobasher, B., Burke, 2002, R. "Ontological user profiles for personalized web search." *In: Proceedings of AAAI 2007 Workshop on Intelligent Techniques for Web Personalization*, pp. 84–91. Vancouver, BC, Canada (2007)

[Spertus *et al.,* 2005] – E. Spertus, M. Sahami, and O. Buyukkokten – "Evaluating similarity measures: a large–scale study in the orkut social network" In KDD '05, pp. 678–684, New York, NY, USA, 2005. ACM.

[Symeonidis *et al.,* 2008] – Symeonidis, P., Nanopoulos, A., Manolopoulos, Y. "Justified recommendations based on content and rating data." *In: WebKDD Workshop on Web Mining and Web Usage Analysis*, 2008

[Takacs *et al.,* 2008] – Takacs G., Pilaszy I., Nemeth B. and Tikk, D., "Matrix Factorization and Neighbor based Algorithms for the Netflix Prize Problem", *Proc. 2nd ACM conference on Recommender Systems (RecSys'08)*, pp.267–274, 2008.

[Toms, 2000] – Toms, Elaine G. "Serendipitous Information Retrieval. " *In DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries*, pp. 17–20. 2000.

[Uschold and Gruninger, 1996] – Uschold, Mike, and Michael Gruninger. "Ontologies: Principles, methods and applications." The knowledge engineering review 11, no. 02, pp. 93–136. (1996)

[Xia et all, 2015] – Xia, P., Liu, B., Sun, Y., & Chen, C. "Reciprocal recommendation system for online dating." *In Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pp. 234–241. ACM. 2015

[Yao, 1995] – Yao, Y.Y.: "Measuring retrieval effectiveness based on user preference of documents." Journal of the Association for Information Science and Technology, #46(2), pp. 133–145 (1995)

[Zhang and Hurley, 2008] – Zhang, M., Hurley, N. "Avoiding monotony: improving the diversity of recommendation lists." *In: RecSys'08: Proceedings of the 2008 ACM conference on Recommender systems*, pp. 123–130. ACM, New York, NY, USA (2008)

[Zhang *et al.,* 2002] – Zhang, Y., Callan, J., Minka, T. "Novelty and Redundancy Detection in Adaptive Filtering." *In: Proceedings of the 25th International ACM SIGIR Conference*, pp. 81–88, 2002

[Zheng and Peltzverger, 2015] – Zheng, J. G. and Peltsverger, S. "Web Analytics Overview," article in Encyclopedia of Information Science and Technology, Third Edition, IGI Global 2015

[Zhou R. *et al.,* 2016] – Zhou, Renjie, Samamon Khemmarat, Lixin Gao, Jian Wan, and Jilin Zhang. "How YouTube videos are discovered and its impact on video views." Multimedia Tools and Applications 75, no. 10, pp. 6035–6058. (2016)

[Zuo *et al.,* 2016] – Zuo, Y., Zeng, J., Gong, M., & Jiao, L. "Tag–aware recommender systems based on deep neural networks." Neurocomputing, #204, pp. 51–60. 2016

[Ding and Liu, 2015] – Ding, Y., & Liu, C. "Exploring drawbacks in music recommender systems: the Spotify case." 2015, http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A896794&dswid=4262 last access 24.12.2016

[eBay, 2013] – Thomas Pinckney – "Graph–based Recommendation Systems at eBay", https://www.youtube.com/watch?v=Tg3dP2fZGSM last access 15.01.2017

[Funk, 2016] – Funk, S., http://sifter.org/~simon/journal/20061211.html "Netflix Update: Try This At Home", last access 09.11.2016.

[Google Analytics, 2016] – https://analytics.google.com/analytics/web last access 09.11.2016

[Howe, M, 2009] – Howe Michael, "Pandora's Music Recommender. A Case Study", I, pp. 1–6. https://courses.cs.washington.edu/courses/csep521/07wi/prj/michael.pdf last access 11.03.2017

[jsoup, 2016] – https://jsoup.org/ last access 15.11.2016

[Lucene, 2016] – http://lucene.apache.org/core/ last access 22.11.2016

[OWA, 2016] – http://www.openwebanalytics.com/ last access 05.11.2016

[OWL, 2012] – https://www.w3.org/OWL/ last access 18.03.2017

[Piwik Study, 2016] – https://piwik.pro/wp–content/uploads/2016/01/Canada_Government_Case–Study.pdf last access 07.11.2016

[Piwik, 2016] – https://piwik.org/ last access 07.11.2016

[Protégé, 2016] – http://protege.stanford.edu/ last access 10.10.2016

[RDF, 2014] – https://www.w3.org/TR/rdf11–concepts/ last access 12.10.2016

[RDFs, 2014] – https://www.w3.org/TR/rdf–schema/ last access 19.02.2017

[sitemapgen4j, 2016] – https://code.google.com/archive/p/sitemapgen4j/ last access 09.10.2016

[w3techs, 2016] – https://w3techs.com/technologies/details/ta–googleanalytics/all/all last access 04.11.2016

[Woopra, 2016] – https://www.woopra.com/ last access 19.10.2016

[WordNet] – https://wordnet.princeton.edu/ last access 09.10.2016

[XML, 2008] – https://www.w3.org/TR/REC–xml/ last access 12.10.2016

[YM, 2016] – https://metrica.yandex.com last access 06.11.2016