UDC 004.738

User Profiling for the Web

Miha Grčar[1], Dunja Mladenič[1], Marko Grobelnik[1]

[1] J.Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
{miha.grcar, dunja.mladenic, marko.grobelnik}@ijs.si

Abstract. This paper addresses a problem of personalized information delivery related to the Web, that is based on user profiling. Different approaches to user profiling have been developed. When the user profiling is used for personalization in the context of Web, we can talk about Web personalization. There are three main groups of approaches: content-based filtering, collaborative filtering and Web usage mining. We provide an overview of them including recent research results in the area with especial emphases on user profiling in the perspective of Semantic Web applications.

1. Introduction

We live in a society in which computers and the Internet are widely used for accessing different kinds of information. Today, it is not enough that computers are capable of performing complex tasks in reasonable time and storing huge amounts of data, they also need to be accessible to a wider community. This includes development of natural and adaptive multimodal interfaces that respond intelligently; the development of semantic-based and context-aware systems to acquire, organize, process, share and use the knowledge embedded in multimedia content; and the construction of physically instantiated or embodied systems that can perceive, understand and interact with their environment, and evolve in order to achieve human-like performance [66]. Personalized information delivery based on user and document profiling is an important step in making computers helpful and accessible to a wider community. In connection to the World Wide Web that greatly contributes to the growing number of computer users, we talk about Web personalization.

Web personalization in the broadest sense of the term denotes the process of personalizing Web sites according to the specific user's profile to achieve more efficient Web browsing. By Web browsing, we mainly refer to the ability of the user to easily find relevant items (contents). User's browsing efficiency is increased by altering the Web sites' structures, and by employing recommender systems to produce user-tailored recommendations.

In addition to constructing a user profile based on observing the user behavior (clicked hyperlinks on the Web page, content of the read text documents), the user can explicitly provide feedback to the system (eg.,

labeling Web page as interesting or not, rating music or video - important in collaborative filtering) or provide virtual examples to the system that will result in refining a part of the user profile.

The ways that are employed for Web personalization are mainly: (i) content-based filtering, (ii) collaborative filtering, and (iii) Web usage mining. They are presented in Section 4, Section 5 and Section 6 respectively. Section 2 gives a brief discussion on user profiling related to the Semantic Web while Section 3 discusses different specifics when dealing with Web data. We conclude the paper by discussing some challenges of the area in Section 7.

2. User Profiling for Semantic Web

As semantic web is gaining popularity and raising hopes of many, we provide a brief discussion on user profiling in the context of semantic web. We can say that semantic web is a web supporting “machines talking to other machines” (instead of “people talking to machines” as in the traditional web sense) by explicitly providing semantic data, such as higher-level information about the web page (eg., functionality of a web service provided in the web page or topic category of the web page content). Although machines could be quite “busy” talking to other machines there still needs to be some space left also for human users in the whole process and there is where the “user profiling” comes into the play.

Technically speaking, semantic web is mainly about the data that are self-explanatory, or in other words, about the data which are annotated in some standard fashion, which allow other efficient computer-to-computer communication for the final purpose of building better services for the end-users. Since in general the data can be in most cases understood in more then one way, especially if we are talking about the more abstract categories, which cannot be annotated very explicitly, one of the possible sources of the annotations (meta-data) may come also from the information about the user. The information about the user can be represented in several ways. Typically, if we are talking about more abstract and aggregated information, we talk about “user profiles” or “user models” which have the main characteristic of being able to generalize the collected data about the user behavior (such as click-stream data for user’s browsing behavior). Such “user-models” are then further used for annotating the data in the way that web services are able to deliver personalized information aiming at increase of the user efficiency when communicating with the computer.

To conclude this short introduction about the relation of user profiling to semantic web, we could say that user profiling is an important source of meta-data about the user perspective about the data understanding. In particular, this enables to compensate differences in the understanding of data semantics by alternative annotation, which is more of the soft nature. The

main goal of using user modeling is in increasing the efficiency of user activities by delivering more personalized information.

3. Dealing with Web Data

3.1. Data Sources

There are several kinds of data that are potential input to the Web personalization pipeline. The data can be divided into four main categories: (i) the data from Web access logs, (ii) the content data, (iii) Web site structure data, and (iv) the demographic data [36]. The following subsections briefly describe each of the categories.

Data from Web access logs. The Web logs are maintained by Web servers and contain information about users accessing sites. Logs are mostly stored simply as text files, each line corresponding to one access (i.e. one request). The most widely used log file formats are, implied by [63], the Common Log File format (CLF) [64] and the Extended Log File format (ExLF) [65]. The latter is customizable, which does not apply to CLF. The Web log contains the following information: (i) the user's IP address, (ii) the user's authentication name, (iii) the date-time stamp of the access, (iv) the HTTP request, (v) the response status, (vi) the size of the requested resource, and optionally (vii) the referrer URL (the page the user "came from") and (viii) the user's browser identification. Of course, the user's authentication name is not available if the site does not require authentication. In the worst case, the only user-identification information included in a log file is his/her IP address. This introduces a problem since different users can share the same IP address and, what is more, one user can be assigned different IPs even in the same session (see Section 3.2).

Content Data. The content data are all the contents that can be accessed by users. Here we are not referring only to textual data but also to images and other multimedia contents that are available to users. Usually we are only dealing with textual contents. Dealing with textual contents has been a widely researched topic in information retrieval and text mining (see Section 4).

Web Site Structure Data. The Web site structure data is prepared by the Web site designer or by the structure generator which is employed at the end of the Web personalization pipeline to (semi)automatically produce user-specific structure for more efficient browsing.

Demographic Data. The demographic data was explicitly given by the users, filling out surveys and questionnaires. Such way of data gathering is usually not well accepted by the users. They tend to skip the fill-out process or provide false information. Thus, this kind of data is usually not available, except in cases, where the user directly benefits from providing correct information.

3.2. Data Preprocessing

The first phase of the Web personalization process is the data preprocessing phase [e.g. 37]. Since the demographic data is usually unavailable, we will exclude this kind of data from the process. The Web site structure, on the other hand, skips the data preparation phase, since the data is already "prepared" by the Web site designer or by the structure generator. This means that the data preparation phase can be divided into two sub-phases: (i) the Web access data preparation sub-phase and (ii) the content data preparation sub-phase.

Web access data preparation. The result of the Web access data preparation sub-phase is a large and sparse user-by-item (content) matrix which goes well with the collaborative filtering methods. For the purposes of Web usage mining, sessions or transactions are identified and stored for further processing.

Data cleaning. Not every access to the content should be taken into consideration. We need to remove accesses to irrelevant items (such as button images), accesses by Web crawlers (i.e. non-human accesses), and failed requests.

Efficient user identification. The user's IP address is but poor user-identification information [e.g. 38, 39]. Many users can be assigned the same IP address and on the other hand one user can have several different IP addresses even in the same session. The first inconvenience is usually the side-effect of intermediary proxy devices and local network gateways. Also, many users can have access to the same computer. The second problem occurs when the ISP is performing load balancing over several proxies. All this prevents us from easily identifying and tracking the user. By using the information contained in the "referrer" and "browser" fields we can distinguish between some users that have the same IP, however, a complete distinction is not possible. Cookies can be used for better user identification. Users can block or delete cookies but it is estimated that well over 90% of users have cookies enabled [40]. Another means of good user identification is assigning users usernames and passwords. However, requiring users to authenticate is inappropriate for Web browsing in general.

Session identification and path completion. Sessions should be detected for the Web usage mining process discussed in Section 6. This operation is carried out using the assumption that if a certain predefined period of time between two accesses is exceeded a new session starts at that point. Sessions can have some missing parts. This is due to the browser's own caching mechanism and also because of the intermediate proxy-caches. The missing parts can be inferred from the site's structure [37].

Transaction identification. Some authors propose dividing or joining the sessions into meaningful clusters, i.e. transactions. Pages visited within a session can be categorized as auxiliary or content pages. Auxiliary pages are used for navigation, i.e. the user is not interested in the content (at the time) but is merely trying to navigate from one page to another. Content pages, on the other hand, are pages that seem to provide some useful contents to the user. The transaction generation process usually tries to distinguish between auxiliary and content pages to produce the so called auxiliary-content transactions (consisting of auxiliary pages up to and including the first content page) and the so called content-only transactions (consisting of only content pages). Several approaches, such as transaction identification by reference length [37] and transaction identification by maximal forward reference [37, 61] are available for this purpose.

Content data preparation. In the content data preparation phase [e.g. 26], the contents are converted into a more suitable representational form (see Section 4.1.1). In addition stop-list, and/or word stemming are used to reduce the dimensionality of such representations. The result of this sub-phase is most likely a large and sparse document-by-term matrix suitable for content-based filtering and also used for solving the sparsity problem in collaborative filtering, as discussed in [41, 43].

4. Content Based User Profiling

Large amount of information available in electronic form and offered to the users brings a number of challenges to the users and to research community. The provided information is often an arbitrary mixture of text, speech, image and video in the same document, potentially distributed over different locations and can be frequently changing especially if published on the Web. An additional challenge (and source of information) comes from the fact that different target public and communities meet on the Web around many different topics. One of the commonly addressed problems is providing help to the users in searching and browsing the Web. Some of the developed systems address this problem based on the content analysis using mostly text from the documents while the others are based on information about document relevancy such as users' rating or behavior of the users. For instance, content analysis is used for providing help to the user in Web

browsing by retrieving documents similar to the already requested documents [28].

This Section gives an overview of the methods for automatic user and document profiling that are based on the content of text documents. There are also manual approaches where a user profile is constructed, usually by the user or domain expert, in a form of rules, filters, scripts, such as filters for sorting incoming e-mails into the user e-mail folders. Manual approaches to user profiling are not addressed in this paper.

4.1. Handling Text Data

Recent developments at the intersection of Information Retrieval [32], Data Mining [14, 15, 34] and Machine Learning [12, 24], Natural Language Processing [23] as well as work in Adaptive Hypermedia [7] offer some novel solutions helping users in making a good and quick selection of information they are interested in. These results with the intensive development of methods using primarily data mining and machine learning techniques on text databases, commonly referred to as Text Mining. Some of the typical aspects of text mining research involve development of models for reasoning about text documents based on words, phrases, linguistic and grammatical properties of text and; extracting information and knowledge from large amounts of text documents.

Data representation. One of the first steps when handling text data is decision on the appropriate representation of text to be used. The frequently used representation is so called word-vector representation (or bag-of-words representation) where all the words from a document are taken in a “bag” while ignoring ordering of words or any structure of the text. When having a set of documents, each document is represented as a word-vector having one component for each of the words that occur in any of the documents from the addressed document collection (see Figure 1). We all agree that there is additional information in text documents that could be used, such as some natural language information about the structure of the sentences, word type and role, position of the words or neighboring words. The question is how much can we gain considering additional information in the data analysis (and what information to consider) and what is the price we have to pay for it? There is currently no established comparison or directions for text document representation that we are aware of. There is some evidence in information retrieval research, that for long documents, considering information additional to the word-vectors is not worth the efforts. There is some work on document profiling that extends the word-vector representation by using word sequences (n-grams) instead of single words [29]. This work suggests that the usage of single words and word pairs as features in the word-vector representation improves classification performance of document profiles generated from short documents.

Selection of words. One of the characteristics of text data is usually a large number of different words that occur in the set of text documents. One of the frequently used approaches to reduce the number of different words is to remove words that occur in the stop-list containing common English words, such as “a”, “the”, “with” or pruning the infrequent words [9, 16, 29]. The other common text pre-processing step that is complementary to stop-word removal is replacing surface form of a word by its stem. It reduces the number of different words using a natural language-specific stemming algorithm, for instance, replacing “lives”, “live”, “lived” by “live”.

Natural language independent approach to reducing the number of different words is based on scoring the words in order to select only the best words [10, 16, 29] or reduce the dimensionality using latent semantic indexing with singular value decomposition [4] or concept decomposition using clustering [11]. Experiments with different numbers of selected features used in text classification [16, 29, 35] indicate that the best results are obtained either using only a small percentage of carefully selected features (up to 10% of all features) or in some case using all the features. Surprisingly good results are obtained using a simple frequency measure in a combination with a “stop-list” [29, 35].

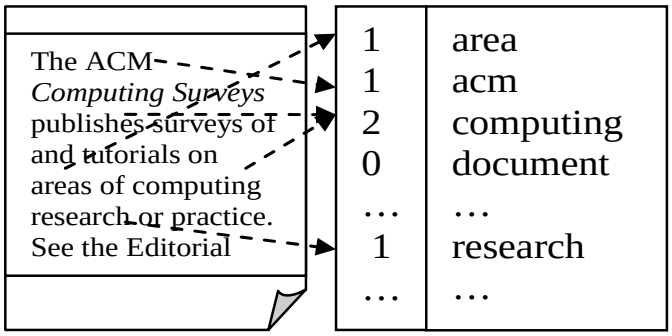

Fig. 1. Illustration of the word-vector document representation using frequency vector.

Algorithms for data analysis. In Information Retrieval, one of the well-established techniques for text document classification is to represent each document using word-vector representation with TFIDF weight (see equation (1)) assign to each word as introduced in [33] and generate a document profile as a sum of all the interesting document vectors. The obtained profile is than used for deciding if the new document is relevant (based on the relevance feedback method Rocchio 1971). Each component of a document word-vector is calculated as the product of Term Frequency (TF) – the

number of times word W occurred in a document and Inverse Document Frequency (IDF). This can be illustrated by the following equation:

$$d^{(i)} = TF(W_i, d)IDF(W_i), where...IDF(W_i) = \log \frac{D}{DF(W_i)} \quad (1)$$

Where D is the number of documents and document frequency DF(W) is the number of documents word W occurred in at least once. The exact formulas used in different approaches may slightly vary but the idea remains the same. A new document is then represented as a vector in the same vector space as the generated model and the distance between these two vectors is measured (usually using the cosine similarity measure) in order to classify the document. This technique is commonly used as a baseline when testing performance of document categorization algorithms and in most cases shows to be inferior to the other document profiling methods.

An extension of TFIDF proposed in [16] called Probabilistic-TFIDF takes into account document representation and was shown to achieve results better than TFIDF and comparable to the Naive Bayesian classifier. The Naive Bayesian classifier and the k-Nearest Neighbor are two classifiers commonly used in machine learning. For instance, the Naive Bayesian classifier was used in [10, 16, 25, 29] and the Nearest Neighbor algorithm was used in [25, 35]. There is a number of other algorithms, with background mainly in machine learning, that have been successfully used for document or user profiling that, such as Decision Rules and Inductive Logic Programming algorithms FOIL and FLIPPER used in [9], Winnow used in [2], Support Vector Machines used in [17, 6]. Support Vector Machines is considered as one of the most successful algorithms for document profiling. Active learning, as one of the approaches aiming at reducing the number of needed labeled documents, was used on text data [30] in a combination with Query by Committee and the Expectation Maximization algorithm.

4.2. User Profiling for Web Browsing

In the content-based approach to user profiling, a profile is based on the content that was requested/visited by the user. One of the main problems with this approach is in capturing different aspects of a complex content such as images or video. In this paper we deal only with text content. Even for text domains, most approaches capture only certain aspects of the content and hope that the main information needed to solve the addressed problem is preserved.

There are different systems that generate the user profile and use it to help the user in Web browsing. Some of them a pro-active and crawl the Web looking for the Web pages that match the user profile. Some of them require feedback from the user, for instance in the form of evaluation of the retrieved pages [3] that is used to update the profile. The others infer the user interests from the browsing behavior [22, 28]. Some approaches combine document

profiling and user profiling, generating a separate user profile for each topic category [1, 3]. User profiling is not necessary connected to Web browsing and can be used in any situation where the user is accessing some content, for instance, in news filtering [20], for composing a personalized newspaper [18], for finding relevant contact person for a specific topic [19] or relevant answer using Frequently Asked Questions [8].

One of the ways of helping the user in Web browsing is by predicting the clicked hyperlinks from the set of Web documents already visited by the user. While the user profile is generated using some of the text data handling approaches (see Section 3.1). This is performed on-line while the user is sitting behind some Web browser and waiting for the requested document. As an example, we here describe a system named Personal WebWatcher [25] that generates a separate model for each user off-line and uses it for highlighting the promising hyperlinks on the requested Web documents. The structure of the system is shown in Figure 2. There is the user on one end and the Web on the other end. Between them is Personal WebWatcher acting as a proxy server. It consists of: proxy that gets http requests from the browser and fetches the requested Web page; adviser that gets the original Web page and extracts the hyperlinks from it and composes the modified Web page by highlighting the promising hyperlinks based on the scores assigned to all the extracted hyperlinks; classifier that threats each extracted hyperlink as an example and uses the induced model of the user interests to assign a score to each of them; LEARNER that gets a collection of visited documents and induces a model of the user's interests based on the Web documents.

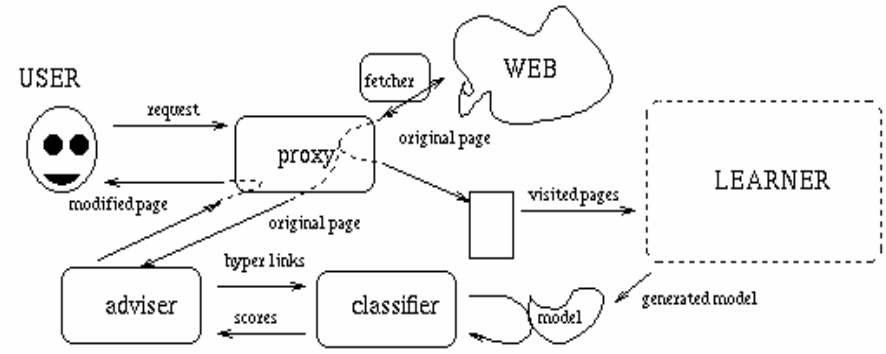

Fig. 2. Structure of Personal WebWatcher. When the system gets the requested Web page, it forwards it to the learner for generating the user profile. At the same time, the original page is equipped with suggestions based on the match between each of the page hyperlinks and the user profile.

Browsing the Web is supported here by highlighting promising (ie., interesting) hyperlinks on the requested Web documents. The assumption is that the interesting hyperlinks are the hyperlinks that are highly probable to be clicked by the user. The problem is defined as predicting clicked hyperlinks

from the set of Web documents visited by the user. All hyperlinks on the visited documents are used for constructing training examples needed by the profile generation algorithm. Each is assigned one of the two class values: positive (the user clicked on the hyperlink) or negative (the user did not click on the hyperlink). Each hyperlink is represented as a kind of small document containing underlined words, words in a window around them and words in all the headings above the hyperlink (the most recent hading for each html-tag for heading size h1 through h6).

As already pointed out, a profile is generated for each user independently of other users. This profile can be further used to compare different users and to share knowledge between them. This sharing of knowledge is related to collaborative approach to user profiling and collaborative intelligent agents [26]. A way of cooperation between different users using the same system for user customized Web browsing is on the model induction level. Namely, even though each user has a separate user profile, they have a similar form. If we could infer from the user profiles some higher-level knowledge that is independent of a specific set of documents, that knowledge could be shared between the users. That would be especially valuable for new users, where only a small set of documents is available for the model induction.

4.3. Document Profiling for Categorization

Automatic categorization of text document is a well known problem that has attracted many researchers. We describe it here on a problem of document profiling based on a large hierarchy of Web documents [29]. Handling a hierarchy of categories (sometimes also referred to as a topic ontology) can be seen as an extension of the usually addressed problem of document categorization into a flat structure of categories, such as the collection of Reuters news articles. In a Web hierarchy (such as, Yahoo! or Open Directory) documents are connected with hyperlinks forming a hierarchical structure with more general categories closer to the root of the hierarchy. Each category is denoted by keywords that appear on the path from the tree root to the node representing the category. More specific category is named by adding a keyword to the name of the more general category directly connected to it (one level higher in the tree). Some nodes at the bottom of the tree contain mostly hyperlinks to actual Web documents, while the other nodes contain mostly or even only hyperlinks to other nodes in the hierarchy.

The goal here is to assign to an arbitrary text document the right category within the given hierarchy as accurate and as fast as possible. The evaluation of the system is based on the list of categories and keywords that are assigned the highest probability. The system architecture proposed in [29] is shown in Figure 3.

In order to handle the hierarchical structure of categories, the whole problem is divided into sub-problems, each corresponding to one of the original categories. For each of the sub-problems, a classifier is constructed using machine learning methods [24] that predicts the probability that a

document is a member of the corresponding category. On each of the sub-problems the Naive Bayesian classifier [24] is used on word-vector document representation, where each feature represents a sequence of words instead of representing a single word. This approach is not limited to Web hierarchy and can be applied on other hierarchies like for instance, thesaurus.

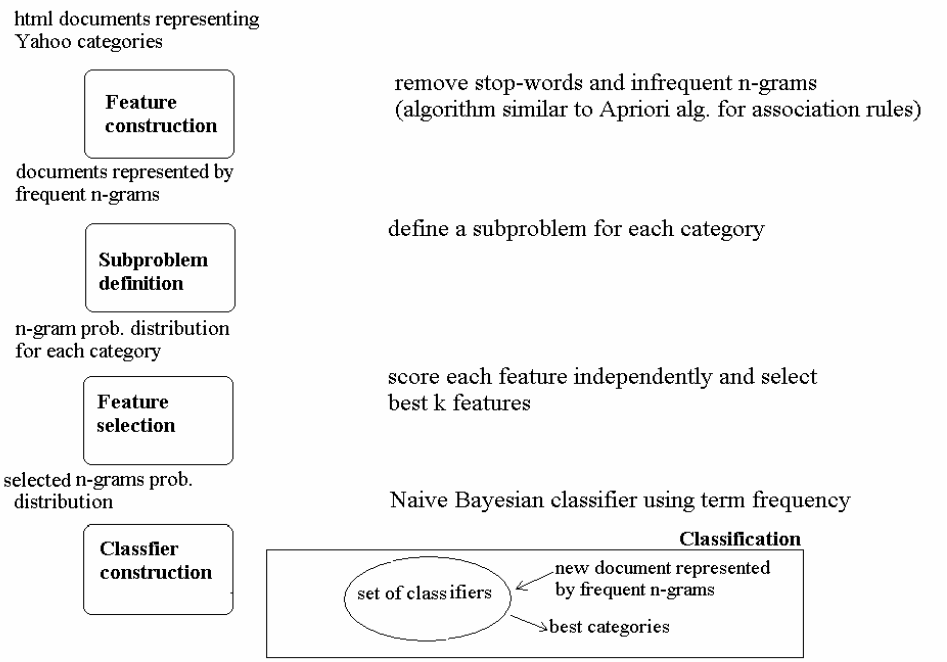

Fig. 3. Architecture of the system for automatic document categorization. First a set of labeled Web documents is processed to get the set of potential feature (words and phrases) to be used in document representation. This phase is named feature construction. Then, all the documents are represented using the constructed features and a feature selection is applied on each of the defined sub-problems. For each of the sub-problems, a classifier is constructed capturing a profile of the corresponding document subset and used later for categorization of a new document.

5. Collaborative User Profiling

Collaborative user profiling as addressed here is using collaborative filtering. It is based on the assumption that "similar users have similar preferences". In other words, by finding users that are similar to the active user and by examining their preferences, the recommender system can (i) predict the active user's preferences for certain items and (ii) provide a ranked list of items which active user will most probably like. Collaborative filtering generally ignores the form and the content of the items and can therefore also be applied to non-textual items. Furthermore, collaborative filtering can detect

relationships between items that have no content similarities but are linked implicitly through the groups of users accessing them. These groups (communities) are formed around a specific user profile.

5.1. Problem Setting for Collaborative Filtering

Before going into any detail, let us first look at the "naive" approach to collaborative filtering that we use in everyday life. If we want to predict how much we will like a movie, we simply look at its average rating at our favorite public list of movie ratings (eg., at IMDb.com). In this case, the prediction is based on ratings given to this movie by other users, and on the assumption that their tastes are similar to ours. Many times, however, we find the movie in question to be over- or underrated by average viewers. This clearly shows that our naive approach is not the most efficient one. Maybe we should ask a friend who we consider to have a similar taste. With this kind of thinking we are moving towards a better approach to collaborative filtering.

Collaborative filtering compares users according to their preferences. Therefore, a database of users' preferences must be available. The preferences can be collected either explicitly (explicit rating) or implicitly (implicit rating). In the first case the user's participation is required. The user explicitly submits his/her rating of the given item. Such rating can, for example, be given as a score on a rating scale from 1 to 5. The implicit ratings, on the other hand, are derived from monitoring the user's behavior. In the context of the Web, access logs can be examined to determine such implicit preferences. For example, if the user accessed the document, he/she implicitly rated it 1. Otherwise the document is assumed to be rated 0 by the user (i.e. "did not visit").

The collaborative filtering process can be divided into two phases: (i) the model generation phase and (ii) the recommendation phase. Algorithms which tend to skip the first phase are the so called memory-based approaches (also referred to as lazy learning approaches or the nearest neighbors algorithms) (see Section 5.2). The preferences database is a huge user-by-item matrix, $R = [r_{i,j}]$, constructed from the data at hand. A matrix element $r_{i,j}$ represents user i's rating of item j. Memory-based approaches search the matrix for relationships between users and/or items. Model-based approaches, on the other hand, use the data from R to build a model that enables faster and more accurate recommendations (see Sections 5.3–5.6). The model generation is usually performed offline over several hours or days.

When dealing with collaborative filtering, two fundamental problems of collaborative filtering have to be taken into account: (i) the sparsity of the data and (ii) the scalability problem. The first problem, which we encounter when R is missing many values, can be partially solved by incorporating other data sources (such as the contents of the items) [41], by clustering users and/or items [42, 43], or by reducing the dimensionality of the initial matrix (see Section 5.3). The last two techniques also counter the scalability problem. This problem arises from the fact that the basic nearest neighbor algorithm

fails to scale up its computation with the growth of the number of users and the number of items. Some of the approaches for countering the two problems are described in Section 5.3.

5.2. Memory-Based Approach to Collaborative Filtering

Approach Description. A straightforward algorithmic approach to collaborative filtering involves finding k nearest neighbors (i.e. the most similar users) of the active user and averaging their ratings of the item in question. Even better, we can calculate weighted average of the ratings, weights being similarity, correlation or distance factors (later on in the text the term similarity is used to denote any of the three measures) between a neighbor-user and the active user [e.g. 42]. We can look at a user as being a feature vector. In this aspect, items that are being rated are features and ratings given by the user to these items are feature values. The following formula can be applied to predict user u's rating of item i:

$$p_{u,i} = \overline{v_u} + \kappa \sum_{j \in Users} w(u, j)(v_{i,j} - \overline{v_j}) \quad (2)$$

Where $w(u_1, u_2)$ is the weight which is higher for more similar, less distant or more correlated users (feature vectors), $\overline{v_u}$ is the mean rating given by user u, $v_{j,i}$ is the rating of item i given by user j, and κ is merely a normalization factor which depends on our choice of weighting.

When representing a user as a feature vector, many of the features have missing values, since not every item was explicitly rated by the user. This fact introduces the sparsity problem which implies that measuring similarity between two feature vectors is not a trivial task. Many times two feature vectors have only a few or no overlapping values at all. When the similarity is computed over only a few values, the similarity measure is unreliable. Furthermore, when there is no overlapping between two vectors, the degree of similarity can not be determined.

Equation (2) was introduced by [44]. If no ratings of item *i* are available, the prediction is equal to the average rating given by user u. This is an evident improvement of the equation that simply calculates weighted average.

Weight Computation. The weights can be defined in many different ways. The most popular ways are presented in this section.

Cosine Similarity. The similarity measure can be based on the cosine of the angle between two feature vectors. This technique was primarily used in information retrieval for calculating similarity between two documents, where

documents were usually represented as vectors of word frequencies. In this context, weights can be defined as given in equation (3).

$$w(u_1,u_2)=\sum_{i\in Items}\frac{V_{u_1,i}\cdot v_{u_2,i}}{\sqrt{\sum_{k\in I_1} v_{u_1,k}^{\ 2}}\sqrt{\sum_{k\in I_2} v_{u_2,k}^{\ 2}}} \quad (\mathbf{3})$$

Pearson Correlation. Alternatively to using cosine similarity, weights can be defined in terms of the Pearson correlation coefficient [44]. Pearson correlation (given in equation (4)) is also used in statistics to evaluate the degree of linear relationship between two variables. It ranges from –1 (a perfect negative relationship) to +1 (a perfect positive relationship), with 0 stating that there is no relationship whatsoever.

$$w(u_1,u_2)=\frac{\sum_{j\in Items}(v_{u_1,j}-\overline{v_{u_1}})(v_{u_2,j}-\overline{v_{u_2}})}{\sqrt{\sum_{j\in Items}(v_{u_1,j}-\overline{v_{u_1}})^2\sum_{j\in Items}(v_{u_2,j}-\overline{v_{u_2}})^2}} \quad (\mathbf{4})$$

Weights Amplification and Inverse User Frequency. We can have good confidence in the computed weight in the case when a lot of overlapping values are available. On the other hand, if there are only few overlapping values, the weight's reliability is questionable. To incorporate the degree of confidence into equation (2), we can lower the weights that are based on only few items and vice versa [e.g. 43]. Additionally, we can amplify weights [42]. This means that we reward weights that are close to 1 and punish those that are close to 0. Another approach for bettering the weights is also the application of the inverse user frequency as described in [42]. The main idea is that universally liked items are less relevant for predictions than those that are popular with a smaller number of people. Therefore, we transform each rating by multiplying it with the inverse user frequency which is defined as log n/n_j, where n_j is the number of users who have rated item j and n is the total number of users.

Default Rating. The problem with the Pearson correlation formula is that only the overlapping ratings can be used for computation. Due to high sparsity of the data, the number of overlapping ratings is rather small. If user A is overlapping with user B in items 1, 2 and 7, and user B is overlapping with user C in items 4, 8 and 12, but users A and C are not overlapping in their rated items, then no relationship can be detected between users A and C. In other words, using Pearson correlation we cannot detect transitive relationships. To avoid this problem, we introduce a slightly modified equation, referred to as default rating. Instead of considering the intersection of available ratings from both users, we now take their union and fill in the missing values with some predefined default value d [42]. At this point we could also consider filling in the user's average rating instead of the constant

d. The missing ratings could also be predicted by averaging the ratings of the items that have similar content. The latter possibility is explored by the so called content-boosted collaborative filtering [41].

5.3. Dimensionality Reduction Techniques for Collaborative User Profiling

In collaborative user profiling, we are initially dealing with a huge user-by-item matrix. Since there can be millions of users and millions of items, the need to reduce the dimensionality of the matrix emerges. The reduction can be carried out by selecting only relevant users (instance selection) and/or by selecting only relevant items (feature selection). Other forms of dimensionality reduction can also be employed, as described later on in this section.

It is shown by some researchers that feature selection, instance selection and other dimensionality reduction techniques not only counter the scalability problem but also result in more accurate recommendations [43, 45, 47]. Furthermore, the sparsity of the data is consequentially decreased.

When reducing the dimensionality, the first possibility that comes to mind is the removal of the users that did not rate enough items to participate in collaborative filtering. From the remaining users, we can randomly choose n users to limit our search for the neighborhood of the active user. This method is usually referred to as random sampling. Also, rarely rated items can be removed for better performance. Still, these relatively simple approaches are usually not sufficient for achieving high scalability and maintaining the recommendation accuracy.

Latent Semantic Analysis (LSA). One of the more sophisticated dimensionality reduction approach is called Latent Semantic Analysis (LSA) [48]. It is based on Singular Value Decomposition (SVD) of the user-by-item matrix. By using linear algebra, a matrix can be decomposed into a triplet, namely $M = U\Sigma VT$. The diagonal matrix Σ holds the singular values of M. If we set all but K largest singular values to zero and thus obtain Σ', we can approximate M as $M' = U\Sigma'VT$. By doing so, we transform our initial high-dimensional matrix into a K-dimensional (low-dimensional) space. The neighborhood of the user can now be determined by transforming the user vector into the low-dimensional space of the approximated matrix and finding k nearest points representing other users. Searching through a low-dimensional space is clearly faster. Furthermore, dimensionality reduction reduces sparsity and captures transitive relationships among users. This results in higher accuracy.

Probabilistic Latent Semantic Analysis (pLSA). On the basis of LSA, Probabilistic Latent Semantic Analysis (pLSA) was presented [46]. pLSA has its roots in information retrieval but can also be employed for collaborative filtering [45]. In a statistical model, an event like “person u ‘clicked on’ item i”

is presented as an observation pair (u, i) (note that in such case we are dealing with implicit ratings). User u and item i "occur" paired with a certain probability: P(u, i). We are in fact interested in the conditional probability of item i occurring given user u: P(i | u). This conditional form is more suitable for collaborative filtering since we are interested in the active user's interests.

The main idea of an aspect model (such as pLSA) is to introduce a latent variable z, with a state for every possible occurrence of (u, i). User and item are rendered independent, conditioned on z: P(u, i) = P(z)P(u | z)P(i | z). P(i | u) can be written as given in the equation (5).

$$P(i \mid u) = \sum_z P(i \mid z)P(z \mid u)\,. \qquad (5)$$

Note that we limit the number of different states of z so that it is much smaller than the number of (u, i) pairs. Let us denote the number of users with N_u, the number of items with N_i, and the number of different states of z with N_z, where $N_z << N_u, N_i$. We can describe the probabilities P(i | u) with $S_1 = N_i \times N_u$ independent parameters. On the other hand, we can summarize the probabilities P(i | z) and P(z | u) with $S_2 = N_i \times N_z + N_u \times N_z$ independent parameters. The dimensionality reduction is evident from the fact that $S_2 < S_1$ (if N_z is small enough). Such latent class models tend to combine items into groups of similar items, and users into groups of similar users. In contrast to clustering techniques (see Section 5.6), pLSA allows partial memberships in clusters (clusters being different states of z).

In equation (5), the probabilities P(z | u) and P(i | z) can be determined by the Expectation Minimization algorithm using various mixture models. To support explicit ratings, we extend pLSA by incorporating rating to our observation pair and thus observing triplets of the form (u, i, r), where r represents a rating score.

The relation of this method to LSA and SVD can be explained by representing the probabilities P(u, i) in the form of a matrix M_p which can be decomposed into three matrices, namely $M_p = U_p \Sigma_p V_p^T$. The elements of these matrices are $u_{i,k} = P(u_i \mid z_k)$, $\sigma_{k,k} = P(z_k)$, $v_{j,k} = P(i_j \mid z_k)$ [40].

Incorporating content into a latent class statistical model. A similar way of dimensionality reductions is to first convert the user-by-item matrix into a user-by-class matrix [43]. Items are classified by using the Naive Bayesian classifier and the Expectation Minimization algorithm. The classification is carried out according to the textual contents of items [49]. Each element of this new matrix is a total of all the items' ratings within a particular class, with respect to the specific user. This way, we can define the neighborhood of the user faster, since the number of classes is much smaller than the number of items. We can also apply instance selection in an elegant way. In a latent class statistical mixture model (e.g. pLSA), a (user, item) pair occurs with a certain probability. More precisely $P(u,i) = \sum_c P(u \mid c)P(i \mid c)P(c)$, that can be transformed as given in equation (6).

$$P(i \mid u) = P(i) \sum_c \frac{P(c \mid u) P(c \mid i)}{P(c)} . \quad (6)$$

The latter depicts the relevancy of user u to item i based on a latent class model. Probabilities P(c | i) have already been determined in the classification phase. Additionally, P(c | u) and P(c) are approximated from the data at hand (see [43]). For the given target item we only need to consider instances (i.e. users) with high probability of occurring paired with the target item.

5.4. Collaborative Filtering as a Classification Task

The collaborative filtering task can also be interpreted as a classification task, classes being different rating scores [50]. Virtually any supervised learning algorithm can be applied to perform classification (i.e. prediction). For each user we train a separate classifier. A train set consists of feature vectors representing items the user already rated, classifications being rating scores from the user. Clearly the problem occurs if our training algorithm cannot handle missing values in the sparse feature vectors. It is suggested by [50] to represent each user by several instances (optimally, one instance for each possible rating score). On a 1–5 rating scale, user A would be represented with 5 instances, namely A-rates-1, A-rates-2, ..., A-rates-5. The instance A-rates-3, for example, would hold ones ('1') for each item that user A rated 3 and zeros ('0') for all other items. This way, we fill in the missing values. We can now use such binary feature vectors for training. To predict a rating, we need to classify the item into one of the classes representing rating scores. If we wanted to predict scores on a continuous scale, we would have to use a regression approach instead of classification.

5.5. Item-Based Collaborative Filtering

All the collaborative filtering approaches we have discussed so far are user-centric in the way that they concentrate on determining the user's neighborhood. Some researchers also considered item-based collaborative filtering [51]. The main idea is to compute item-item similarities (according to the users' ratings) offline and make use of them in the online phase. To predict user u's rating of item i, the online algorithm computes a weighted sum of the user u's ratings over k items that are most similar to item i. The main question in this approach is how to evaluate item-item similarities to compute a weighted sum of the ratings. Item-item similarities can be computed using the techniques for computing user-user similarities, described in Section 5.

The winning technique, according to [51], is the so called adjusted cosine similarity measure. This is a variant of cosine similarity which incorporates the fact that different users may have different rating scales. The similarity

measures are then used as weights for calculating a weighted sum of k nearest items.

5.6. Some Other Approaches

Let us briefly summarize some other techniques. Interested reader should consider the appropriate additional reading.

Horting. Horting is a graph-theoretic approach to collaborative filtering [52]. It involves building a directed graph in which vertices represent users and edges denote the degree of similarity between them. If we are trying to predict user u's rating of item i, we need to find a directed path from user u to a user who has rated item i. By using linear transformations assigned to edges along the path, we can predict user u's rating of item i. No other user along this path rated item i. This means that horting also explores transitive relationships between users.

Clustering Techniques. Bayesian and non-Bayesian clustering techniques can be used to build clusters (or neighborhoods) of similar users [42, 43, 45]. The active user is a member of a certain cluster. To predict his/her rating of item i, we compute the average rating for item i within the cluster that the user belongs to. Some such methods allow partial membership of the user in more than one cluster. In such case, the predicted rating is summed over several clusters, weighted by the user's participation degree. Clustering techniques can also be used as instance selection techniques (instances being users) that are used to shrink the candidate set for the k nearest neighbors algorithm.

Bayesian Networks. Bayesian networks with a decision tree at each node have also been applied to collaborative filtering [42, 53]. Nodes correspond to items, and states of each node correspond to possible rating scores. Conditional probabilities at each node are represented in a form of decision trees in which nodes again are items, edges represent preferences, and leaves represent possible states (i.e. rating scores). Bayesian networks are build offline over several hours or even days. This approach is not suitable in systems that need to update rapidly and frequently.

6. Web Usage Mining for User Profiling

Web usage mining differs from collaborative filtering in the fact that we are not interested in explicitly discovering user profiles but rather usage profiles. When preprocessing a log file we do not concentrate on efficient identification of unique users but rather try to identify separate user sessions. These

sessions are then used to form the so called transactions (see [37]). In the following stage, Web usage mining techniques are applied to identify frequent item-sets, sequential patterns, clusters of related pages and association rules (see Sections 6.2 and 6.3). Web usage mining can be used to support dynamic structural changes of a Web site in order to suit the active user, and to make recommendations to the active user that help him/her in further navigation through the site he/she is currently visiting. Furthermore, recommendations can be made to the site administrators and designers, regarding structural changes to the site in order to enable more efficient browsing. In the case of implementing Web usage mining system in the form of a proxy server, predictions about which pages are likely to be visited in near future can be made, based on the active users' behavior. Such pages can be pre-fetched to reduce access times.

6.1. Web Usage Mining vs. Collaborative Filtering

Web usage mining shows some similarities to collaborative filtering. If we consider pages to be items and we are able to efficiently identify users, we can perform collaborative filtering in order to provide the active user with recommendations about which pages he/she should also visit. Furthermore, we can point out links to pages that the active user will probably navigate to next. This approach, however, has several drawbacks. Each time the user accesses the site he/she may have different browsing goals. The user might prefer recommendations that focus on his current interest. Furthermore, the information about the sequential order of accesses is discarded in collaborative filtering. It is shown in [54] that this piece of information is significant for predictive tasks such as pre-fetching while it is less desirable for the recommendation tasks of collaborative filtering. Another problem arises when an efficient tracking mechanism based on user authentication and/or cookies is not available. In this case it is probably better to perform a variant of Web usage mining.

Since the user may have different browsing goals each time he/she accesses the site and since sessions are easier to identify in log files than users, sessions can be used as instances (instead of users). Each session is thus represented in the form of a feature vector as follows: $s = (w_1, w_2, ..., w_n)$, where weight w_k is determined by the degree of the user's interest in the k-th page during session s, as described in Section 6.2.

We are now dealing with feature vectors, features being items (pages), just as in collaborative filtering. However, in this case vectors represent sessions and not users, which distinguish Web usage mining from collaborative filtering.

6.2. Algorithmic Approach

In this section we describe the algorithm for Web usage mining, presented in [55]. In the data preprocessing phase we extract a set of transactions [37]. Each transaction can be represented as a feature vector, features being pages and feature values being weights denoting the degree of the user's interest in a certain page during the transaction: $t = (w_1, w_2, ..., w_n)$. Weights w_k can be defined in different ways. One of the possibilities is to represent them by the amount of time the user spends on a page. Note that the Web server has no exact notion of the time spent on a page. The duration of the visit can only be estimated from the time difference between two consecutive accesses. This approach seems reasonable, since it tends to weight content pages higher. However, it was observed in [55] that one long access can completely obscure the importance of other relevant pages. If we are dealing with transactions that do not contain navigational pages since these were filtered out, it is probably better to use other approaches. In such case, weights can be defined by the number of times a page was visited during the transaction. We can also simply use binary values stating "the page was visited at least once" and "the page was not visited".

Once a vector representation of transactions is obtained, we need to define a distance measure $d(t_1, t_2)$ between two vectors for the purpose of clustering the transactions. Cosine similarity measure can be used; the distance is in this case computed as $d(t_1, t_2) = 1 - \cos\theta(t_1, t_2)$, where θ is the angle between the two vectors t_1 and t_2. Another possibility is to use the Euclidean distance, computed as $d(t_1,t_2) = \sqrt{\sum_{i=1,...,n}(w_i^{(t_1)} - w_i^{(t_2)})^2}$. We can also define the distance measure by counting the number of overlapping non-zero weights n_{ovrl} in both vectors; $d(t_1, t_2) = 1 - n_{ovrl}/n$. The latter measure is used when comparing the active user's current session to the cluster medians, as described later on in the text.

In the next step, an unsupervised clustering algorithm is employed to discover different usage profiles. There are several well-known approaches that can be employed for this tasks, such as the leader algorithm, k-means algorithm, fuzzy membership approaches, Bayesian clustering, etc. Once clusters are obtained, a median transaction can be computed for each cluster: $\bar{t} = (\overline{w_1}, \overline{w_2}, ..., \overline{w_n})$. The main characteristics of a cluster are evident from its median transaction. Pages with higher median weights contribute more to the nature of the cluster.

The active user's current session (referred to as an active session) is maintained in the form of a vector. Each time the user requests a new page, the vector is updated and compared to cluster medians in order to find the cluster in which the user's current browsing behavior can be categorized. Not all similarity measures are equally successful in this task. In [55] weights are computed by counting the number of overlapping non-zero weights (denoted

by n_{ovrl}) in both vectors (the active session vector and cluster median $\bar{t}$) and applying the following distance formula: $d(s_a, \bar{t}) = 1 - n_{ovrl} / n$.

In the following step, medians and thus clusters that are very similar to the active session (this means that the distance is below some predetermined threshold) are discovered. Pages in these clusters that have high median weights and are not contained in the active session are then recommended to the user. An additional weighting can be done to reward pages that are farther away from the active session with respect to the site's structure. These recommendations tend to be more interesting, since they are providing shortcuts to other (distant) sections of the site. Other more sophisticated methods for providing interesting recommendations have also been discussed [59].

Some authors argue that clustering based on distance measures is not the most prospective approach [58]. They state that the similarity (distance) computation is not a trivial task since vector representations are usually not good behavioral indicators when it comes to Web transactions. They propose a slightly different approach involving association rules discovery. These approaches are discussed in the next section.

6.3. Association Rules Discovery in Web Usage Mining

Some authors find the association rules discovery approach to be more prospective than the approach discussed in Section 6.2. After transactions are detected in the preprocessing phase, frequent item-sets are discovered using the A-priori algorithm [e.g. 60]. The support of item-set I is defined as the fraction of transactions that contain I and is denoted by $\sigma(I)$. Given two item-sets X and Y, the association rule can be expressed as $<X \Rightarrow Y, \sigma_r, \alpha_r>$, where σ_r is the support of $X \cup Y$ and α_r is the confidence of the rule given by $\sigma_r/\sigma(X)$.

Hypergraph Representation of Frequent Item-sets. Frequent item-sets and their corresponding association rules are represented in the form of a hypergraph. A hypergraph is an extension of a graph where each hyperedge can connect more than two vertices. A hyperedge connects URLs within a frequent item-set. Each hyperedge is weighted by the averaged confidence of all the possible association rules formed on the basis of the frequent item-set that the hyperedge represents. The hyperedge weight can be perceived as a degree of similarity between URLs (vertices). Since the hyperedge weight can be interpreted as a degree of similarity between vertices, the hypergraph can be partitioned into clusters using the hypergraph partitioning methods [e.g. 62].

Clusters formed in this way are examined to filter out vertices that are not highly connected to the rest of the vertices in the cluster. The measure for determining the degree of connectedness between vertex v and cluster c is defined as follows:

$$conn(v,c) = \frac{|\{edge : edge \subseteq c, v \in edge\}|}{|\{edge : edge \subseteq c\}|}. \quad (7)$$

Equation (7) measures the percentage of edges within the cluster that vertex v is associated to. A high degree of connectedness indicates that v is connected to many other vertices in the partition and is thus highly connected to the partition.

Figure 4 shows a simple hypergraph consisting of two hyperedges corresponding to two item-sets {url1, url2, url3} and {url1, url2, url4, url5}. Let us say, for example, that all possible association rules derived from the first item-set have the following confidence values (noted above the "implies" symbol): $\{url_1\} \overset{0.8}{\Rightarrow} \{url_2, url_3\}$, $\{url_1, url_2\} \overset{0.4}{\Rightarrow} \{url_3\}$, $\{url_1, url_3\} \overset{0.6}{\Rightarrow} \{url_2\}$, $\{url_2\} \overset{0.4}{\Rightarrow} \{url_1, url_3\}$, $\{url_2, url_3\} \overset{0.8}{\Rightarrow} \{url_1\}$ and $\{url_3\} \overset{0.6}{\Rightarrow} \{url_1, url_2\}$. In this case the average confidence value – and thus the hyperedge weight – is 0.6. In this example, the other hyperedge has a weight of 0.1. If we now wish to partition this hypergraph into two clusters, we need to cut one or more hyperedges so that there are no interconnections between the two clusters. The cost of the partitioning is the sum of the weights of all the hyperedges that are cut in the process. We need to minimize this cost to make the partitioning reasonable. If we cut, for example, between vertices url_1 and url_2, the cost is 0.6 + 0.1 = 0.7. The lowest cost is achieved by cutting the hyperedge between url_2 and url_4, or between url_4 and url_5 (the cost is 0.1). The first cut gives us a more balanced partitioning, so it is best to cut the hyperedge between url_2 and url_4 (the dashed line). This gives us two clusters, namely {url_1, url_2, url_3} and {url_4, url_5}. In the first cluster, url_1 and url_2 are more strongly connected to the cluster than url_3 (see the definition of the connectedness function, conn(v, c)). Whether we filter url_3 out or not, depends on our choice of the threshold.

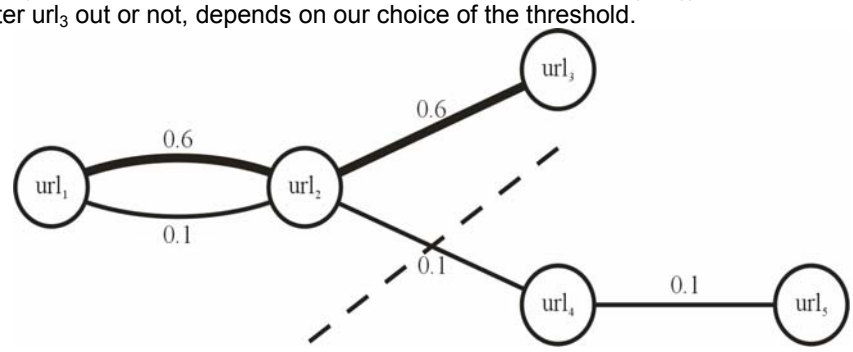

Fig. 4. A simple hypergraph consisting of two hyperedges representing two frequent item-sets

To maintain the active session, a sliding window is used to capture the most recent user's behavior. The window size is determined by the average transaction size, estimated during the pre-processing phase. At each step, the

partial active session is matched with the usage clusters. Each cluster is represented in the form of a vector: c = ($u_1^{(c)}$, $u_2^{(c)}$, ..., $u_n^{(c)}$), where u_k is the weight associated with the k-th URL (url_k) in the following way:

$$u_k^{(c)} = \begin{cases} conn(url_k, c), url_k \in c \\ 0, otherwise \end{cases}$$

The partial active session is represented as a binary vector s = (s_1, ..., s_n), where s_k = 1 if the user accessed url_k in this session, and s_k = 0, otherwise. The next step is to compute the cluster-session matching score, match(s, c). In [58] the following equation is presented:

$$match(s,c) = \frac{\sum_k u_k^{(c)} s_k}{|s| \sqrt{\sum_k (u_k^{(c)})^2}}$$

After matching clusters are determined, the final step is to compute a recommendation score for URL u contained in a matching cluster, according to session s:

$$\mathrm{Re}c(s,u) = \sqrt{conn(u,c) \cdot match(s,c)} \cdot ldf(s,u)$$

where an additional weighting is assigned to reward pages that are farther away from the active session with respect to the site's structure (incorporated with the so called link distance factor, ldf(s, u) [see 58]). URLs with high recommendation scores are recommended to the active user.

Incorporating sequential order of accesses. Sequential order of the accesses in transactions is an important piece of information, mainly for the pre-fetching task. The association rules discovery approach to Web usage mining can be extended with the ability to detect frequent traversal patterns (termed large reference sequences) rather than frequent item-sets [61]. Other steps of this approach are modified accordingly, but are similar to the steps of the described approach using hypergraph representation.

7. Challenges

As already said, semantic web is about producing semantic annotations and meta-data for the data which are used for various applications. User profiles in the form of models serve as a source of annotations for softer kinds of data, where not only one semantics (or understanding) is sensible. User models are built by automatic or semi-automatic means using machine learning and data mining methods. One of the most important challenges for building user-models is an efficient semi-automatic mode where only limited amount of human time is available for providing answers to different questions.

In real-world situations we would like to reduce the amount of that manual work needed by most of the approaches to automatic document filtering,

categorization, user profiling, information extraction, text tagging. For instance, in document profiling for automatic document categorization, we start with a set of documents where each document is assigned to some categories based on its content such as in Yahoo collection. Using unlabeled data and bootstrapping learning are two directions that enable important reduction in the needed amount of hand labeling and can potentially be useful for semantic Web applications.

In document categorization using unlabeled data, we need a small number of labeled documents and a large pool of unlabeled documents, eg., classify an article in one of the 20 News groups, classify Web page as student, faculty, course, project,... The approach proposed by [30, 31] can be described as combining Expectation Maximization and the Naive Bayesian classifier as follows. First, train a classifier with only labeled documents and use the trained classifier to assign probabilistically-weighted class labels to all unlabeled documents. Then, train a new classifier using all the documents and iterate until the classifier remains unchanged. It can be seen that the final result heavily depends on the quality of the labels assigned to the small set of hand labeled data, but it is much easier to hand label a small set of examples with a good quality than a large set of examples with medium quality.

Bootstrap learning to classify Web pages is based on the fact that most of the Web pages have some hyperlinks pointing to them. Using that we can describe each Web page either by its content or by the content of the hyperlinks that point to it. First, a small number (eg., 12 documents) of documents is labeled and each is described using the two description. One classifier is constructed from each description independently and used to label a large set of unlabeled documents. A few of that documents for which the prediction was the most confident are added to the set of the labeled documents and the whole loop is repeated. In this way we start with a small set of labeled documents enlarging it through the iterations and hoping that the initial labels were a good coverage of the problem space. This approach was proposed in [5] and supported by the computational learning theory in the proposed Co-Training theorem.

Recent potentially relevant work includes also mining the extracted data [13], where Information Extraction is used to automatically collect information about different companies from the Web. Data Mining methods are then used on the extracted data. As Web documents are naturally through the hyperlinks organized in a graph structure, there are also research efforts on using that graph structure to improve document categorization [9], to improve Web search and visualization of the Web. Usefulness of these methods in semantic Web context is a matter of future research.

8. Acknowledgements

This work was supported by the Slovenian Research Agency and the IST Programme of the European Community under SEKT Semantically Enabled

Knowledge Technologies (IST-1-506826-IP), NeOn Lifecycle Support for Networked Ontologies (IST-4-027595-IP) and PASCAL Network of Excellence (IST-2002-506778). This publication only reflects the authors' views.

9. REFERENCES

1. Ackerman, M., Billsus, D., Gaffney, S., Hettich, S., Khoo, G., Kim, D.J., Klefstad, R., Lowe, C., Ludeman, A., Muramatsu, J., Omori, K., Pazzani, M.J., Semler, D., Starr, B., Yap, P., 1997. Learning Probabilistic User Profiles, AI magazine, 47-56, Vol. 18, no. 2.
2. Armstrong, R., Freitag, D., Joachims, T., Mitchell, T., 1995. WebWatcher: A Learning Apprentice for the World Wide Web, AAAI 1995 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, Stanford.
3. Balabanovic, M., Shoham, Y., 1997. FAB: Content-based collaborative recomender. Communic. ACM 40, 3, 66-72.
4. Berry, M.W., Dumas, S.T., O'Brien, G.W., 1995. Using linear algebra for intelligent information retrieval, SIAM Rewiev, Vol.37, No. 4, 573-595.
5. Blum, A, Mitchell, T., 1998. Combining Labeled and Unlabeled Data with Co-training, In COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers.
6. Brank, J., Grobelnik, M., Milić-Frayling, N., Mladenić, D. 2002. Feature selection using support vector machines. Proceedings of the 3rd International Conf. on Data Mining Methods and Databases for Engineering, Finance, and Other Fields.
7. Brusilovsky, P., Kobsa, A., Vassileva, J. (eds.) 1998. Adaptive Hypertext and Hypermedia, Kluwer Academic Publishers.
8. Burke, R., Hammond, K., Kulyukin, V., Lytinen, S., Tomuro, N., Schoenberg, S., 1997. Question Answering from Frequently-Asked Question Files: Experiences with the FAQ Finder System. AI Magazine, 18(2), pages 57-66.
9. Cohen, W., Singer, Y., 1999. Context-sensitive learning methods for text categorization, in ACM Transactions on Information Systems, v17, 171-173 .
10. Craven, M., Slattery, S. 2001. Relational Learning with Statistical Predicate Invention: Better Models for Hypertext. Machine Learning, 43(1-2): 97-119.
11. Dhillon, I.S., Modha, D.S., 2001. Concept decomposition for large sparse text data using clustering, Machine Learning , Vol. 42, No. 1, 143-175.
12. Duda, R. O., Hart, P. E. and Stork, D. G. 2000. Pattern Classification 2nd edition, Wiley-Interscience
13. Ghani, R., Jones, R., Mladenic, D., Nigam, K., Slattery, S., 2000. Data Mining on Symbolic Knowledge Extracted from the Web, In KDD-2000 Workshop on Text Mining, 2000.
14. Hand, D.J., Mannila, H., Smyth, P. 2001) Principles of Data Mining (Adaptive Computation and Machine Learning), MIT Press.
15. Hastie, T., Tibshirani, R. and Friedman, J. H. 2001. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer Series in Statistics, Springer Verlag.
16. Joachims, T., 1997. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization, Proc. of the 14th International Conference on Machine Learning ICML97, 143-151.
17. Joachims, T, 2002. Learning to Classify Text using Support Vector Machines, Dissertation, Kluwer Academic Publishers.

18. Kamba, T., Sakagami, H., Koseki, Y., 1997. ANATAGONOMY: a personalized newspaper on the World Wide Web, International Journal Human-Computer Studies, 46, 789-803.
19. Krulwich, B., Burkey, C., 1996. The ContactFinder agent: Answering bulletin board questions with referrals, In Proceedings of National Conference on Artificial Intelligence (AAAI-96), 10-15.
20. Lang, K., 1995. News Weeder: Learning to Filter Netnews, Ken Lang. NewsWeeder: Learning to filter netnews. In A. Prieditis and S. Russell, editors, Proceedings of the 12th International Conference on Machine Learning (ICML-95), pp, 331-339. Morgan Kaufmann.
21. Lewis, D.D., Schapire R.E., Callan, J.P., Ron Papka, R., 1996. Training Algorithms for Linear Text Classifiers, Proc. of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)}, 298-306.
22. Lieberman, H., 1995. Letizia: An Agent that Assists Web Browsing, In Proceedings of International Joint Conference on Artificial Intelligence IJCAI95.
23. Manning, C.D., Schutze, H., 2001. Foundations of Statistical Natural Language Processing, The MIT Press, Cambridge, MA.
24. Mitchell, T.M. 1997. Machine Learning. The McGraw-Hill Companies, Inc..
25. Mladenic, D., 1996. Personal WebWatcher: Implementation and Design, Technical Report IJS DP-7472.
26. Mladenic, D. 1999. Text-learning and related intelligent agents. IEEE EXPERT, Special Issue on Applications of Intelligent Information Retrieval.
27. Mladenic, D. and Grobelnik, M. 1999. Feature selection for unbalanced class distribution and Naive Bayes, Proceedings of the 16th International Conference on Machine Learning ICML-99, Morgan Kaufmann Publishers, San Francisco, CA, 258 - 267.
28. Mladenic, D. 2002. Web browsing using machine learning on text data, In (ed. Szczepaniak, P. S.), Intelligent exploration of the web, 111, Physica-Verlag, 288–303.
29. Mladenic, D., Grobelnik, M. 2003. Feature selection on hierarchy of web documents. Journal of Decission support sysems, 35, 45-87.
30. Nigam, K., McCallum, A., 1998. Pool-Based Active Learning for Text Classification, Working notes of Learning from Text and the Web, Conference on Automated Learning and Discovery CONALD-98, 1998.
31. Nigam, K., McCallum, A., Thrun, S., and Mitchell, T., 2001. Text Classification from Labeled and Unlabeled Documents using EM, Machine Learning Journal.
32. Rijsberg, C. J., van 1979), Information Retrieval, Butterworths.
33. Salton, G., Buckley, C., 1987. Term Weighting Approaches in Automatic Text Retrieval, Technical report, COR-87-881, Department of Computer Science, Cornell University.
34. Witten, I.H., Frank, E., 1999) Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann.
35. Yang, Y., Pedersen, J.O., 1997. A Comparative Study on Feature Selection in Text Categorization, Proc. of the 14th International Conference on Machine Learning ICML97, 412-420.
36. Eirinaki, M., Vazirgiannis, M. Web Mining for Web Personalization; ACM Transactions on Internet Technology, Vol. 3, No. 1, 1–27, 2003.
37. Cooley, R., Mobasher, B., Srivastava, J. Data Preparation for Mining World Wide Web Browsing Patterns; Journal of Knowledge and Information Systems, Vol. 1, No. 1, 5–32, 1999.

38. Pitkow, J. In Search for Reliable Usage Data on the WWW; Proceedings of the Sixth International WWW Conference, 1997.
39. Rosenstein, M. What is Actually Taking Place in Web Sites: E-Commerce Lessons from Web Server Logs; ACM Conference on Electronic Commerce, 2000.
40. Baldi, P., Frasconi, P., Smyth, P. Modelling the Internet and the Web; 171–209, ISBN: 0-470-84906-1, 2003.
41. Melville, P., Mooney, R.J., Nagarajan, R. Content-Boosted Collaborative Filtering for Improved Recommendations; Proceedings of the Eighteenth National Conference on Artificial Intelligence, 187–192, 2002.
42. Breese, J.S., Heckerman, D., Kadie, C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering; Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, 1998.
43. Zeng, C., Xing, C., Zhou, L. Similarity Measure and Instance Selection for Collaborative Filtering; Proceedings of the Twelfth International World Wide Web Conference, 2003.
44. Resnick, P. Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering for Netnews; Proceedings of CSCW '94, 1994.
45. Hoffman, T. Latent Semantic Models for Collaborative Filtering; ACM Transactions on Information Systems, Vol. 22, Issue 1, 89–115, 2004.
46. Hoffman, T. Probabilistic Latent Semantic Analysis; Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, 1999.
47. Yu, K., Xu, X., Ester, M., Kriegel, H. Selecting Relevant Instances for Efficient and Accurate Collaborative Filtering; Proceedings of the Tenth International Conference on Information and Knowledge Management, Session: Collaborative Filtering and Algorithms, 239–246, 2001.
48. Deerwester, S., Dumais, S.T., Harshman, R. Indexing by Latent Semantic Analysis; Journal of the Society for Information Science, Vol. 41, Issue 6, 391–407, 1990.
49. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T. Text Classification from Labeled and Unlabeled Documents Using EM; Machine Learning, Vol. 39, Issue 2–3, 103–134, 2000.
50. Billsus, D., Pazzani, M.J. Learning Collaborative Information Filers; Proceedings of the Fifteenth International Conference on Machine Learning, 1998.
51. Sarwar, B., Karyps, G., Konstan, J., Riedl, J. Item-Based Collaborative Filtering Recommendation Algorithms; Proceedings of the Tenth International Conference on World Wide Web, 285–295, 2001.
52. Aggarwal, C.C., Wolf, J.L., Wu, K., Yu, P.S. Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering; Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 201–212, 1999.
53. Chickering, D.M., Heckerman, D., Meek, C.A Bayesian Approach to Learning Bayesian Networks with Local Structure; Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, 1997.
54. Mobasher, B., Dai, H., Luo, T., Nakagawa, M. Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks; Proceedings of the IEEE International Conference on Data Mining, 2002.
55. Yan, T.W., Jacobsen, M., Garcia-Molina, H., Dayal, U. From User Access Patterns to Dynamic Hypertext Linking; Proceedings of the Fifth International World Wide Web Conference, 1996.

56. Srivastava, J., Cooley, R., Deshpande, M., Tan, P. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data; SIGKDD Explorations, Vol. 1–2, 2000.
57. Nasraoui, O., Frigui, H., Joshi, A., Krishnapuram, R. Mining Web Access Log Using Relational Competitive Fuzzy Clustering; Proceedings of the International Conference on Knowledge Capture, 202–208, 2001.
58. Mobasher, B., Cooley, R., Srivastava, J. Creating Adaptive Web Sites Through Usage-Based Clustering of URLs; Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange, 1999.
59. Cooley, R., Tan, P., Srivastava, J. Discovery of Interesting Usage Patterns from Web Data; Proceedings of the International Workshop on Web Usage Analysis and User Profiling; 163–182, 1999.
60. Agrawal, R., Imielinski, T., Swami, A. Mining Association Rules between Sets of Items in Large Databases; Proceedings of the 1993 ACM SIGMOD Conference, 1993.
61. Chen, M., Park, J.S., Yu, P.S. Data Mining for Path Traversal Patterns in a Web Environment; Proceedings of the 16th International Conference on Distributed Computing Systems, 385, 1996.
62. Han, E., Karyps, G., Kumar, V., Mobasher, B. Clustering Based on Association Rule Hypergraphs; Proceedings of Workshop on Research Issues in Data Mining and Knowledge Discovery, 1997
63. Netcraft: Netcraft Web Server Survey; http://www.netcraft.com/survey/archive.html
64. W3C: Logging Control in W3C httpd; http://www.w3.org/Daemon/User/Config/Logging.html
65. W3C: Extended Log File Format; W3C Working Draft WD-logfile-960323, http://www.w3.org/TR/WD-logfile.html
66. FP6, IST, Knowledge and interface technologies; http://www.cordis.lu/fp6/ist.htm

Miha Grcar is a software developer and a researcher in computer science, currently working at the Jozef Stefan Institute in Ljubljana (http://kt.ijs.si). He is also a former C.E.O. of a small software company that cooperated with Hewlett-Packard for several years, developing components for their imaging library. He is experienced in several development environments, coding mostly in .NET's C# and C++, and is familiar with many popular programming technologies. His research focus is on machine learning (mostly text learning), data mining (mostly text mining), and user/usage profiling (with emphasis on collaborative filtering).

Dunja Mladenic is an expert on study and development of Machine Learning, Data Mining and Text Mining techniques and their application on real-world problems. Her current research focuses on data analysis, with particular interest in learning from Text and the Web. She is associated with the Department of Knowledge Technologies of the J. Stefan Institute since 1992. She got her MSc and PhD in Computer Science at University of Ljubljana in 1995 and 1998 respectively. She was a visiting researcher at School of Computer Science, Carnegie Mellon University, USA in 1996-1997 and in 2000-2001. She has published papers in refereed conferences and journals, served in the program committee of international conferences and organized international events in the area of Text Mining, Link Analysis and Data Mining.

Dunja Mladenic is the Slovenian representative in EC Enwise STRATA ETAN Expert Group ”Promoting women scientists from the Central and Eastern European countries and the Baltic States to produce gender equality in science in the wider Europe”. She is co-editor of the book “Data Mining and Decision Support: Integration and Collaboration”, Kluwer Academic Publishers, 2003 and “Web Mining: from Web to Semantic Web”, Lecture notes in AI, Springer, 2004.

Marko Grobelnik is an expert in analysis of large amounts of complex data with the purpose to extract useful knowledge. In particular, the areas of expertise comprise: Data Mining, Text Mining, Information Extraction, Link Analysis, and Data Visualization as well as more integrative areas such as Semantic Web, Knowledge Management and Artificial Intelligence. Apart from research on theoretical aspects of unconventional data analysis techniques, he has valuable experience in the field of practical applications and development of business solutions based on the innovative technologies. Marko was employed as a researcher first, at the Computer Science Department at University of Ljubljana and later at the Department of Knowledge Technologies at J.Stefan Institute, Ljubljana, Slovenia. His main achievements are from the field of Text-Mining, having a leading role at research and development projects funded by European Commission, having projects with industries including Slovenian publishing house, Slovenian National and University Library and companies such as Microsoft Research. He has published papers in refereed conferences and journals, served in program committee of international conferences and organized a series of international events in the area of Text Mining, Link Analysis and Data Mining.