USER MODELING ON THE WEB

An Exploratory Review of Recommendation Systems

By Olav Bjørkøy
(olavfrih@stud.ntnu.no)

Supervisor: Asbjørn Thomassen
NTNU, Trondheim, December 19, 2010
TDT4500 Intelligent Systems, Specialization Project

Abstract

Information overload occurs when an application presents more information than a user is able to consume. This problem is especially visible on the World Wide Web, where the sheer wealth and scope of information quickly strains the user experience.

The problem becomes even more prominent in applications designed to aggregate content from different sources. Whether the content is written by multiple authors, pertain to varying topics, or is of fluctuating quality, presenting this content in a coherent and interesting way to each individual user is a major challenge.

This paper looks at how adaptive content can help mitigate the problem of information overload in content aggregating web applications. Our tool for the job is *user modeling* — methods in Artificial Intelligence (AI) that create digital representations of users, so that content may be adapted on a personal level.

Specifically, this paper explores the field of *recommender systems*. These are methods designed to estimate just how interesting items of some sort will be to each user. Our survey will investigate the most prominent and novel approaches to user modeling, in the context of improving the content displayed by an online feed reader. We explore the inferential competence, provided features, and the consequences for the user interface of each modeling method. The result is a thorough exploration of what user modeling means for the Web.

Preface

This work on *user modeling on the web* is part of my specialization project at the Norwegian University of Science and Technology (NTNU). My specialization is in the field of *intelligent systems*, at the Department of Computer and Information Science (IDI), faculty of Information Technology, Mathematics and Electrical Engineering (IME).

I would like to thank my supervisor, assistant professor Asbjørn Thomassen, for valuable guidance and feedback throughout the process. In addition, thanks are in order for my fellow students Kjetil Valle and Kim Joar Bekkelund, who helped me formulate my thoughts and provided feedback on the work represented by this document.

To limit the scope of an already extensive topic, this document assumes a basic knowledge of set theory, graph theory and fundamental concepts in artificial intelligence on behalf of the reader.

Trondheim, December 19th, 2010
Olav Bjørkøy

Contents

1 Introduction 9

The Problem 9

Possible Solutions 10

Objectives 11

2 Information Overload 13

Clues From Interaction Design 14

Online Overload 15

Adaptive Content 17

Solving the Problem 19

3 User Modeling 21

Recommender Systems 21

Estimating Ratings 22

Machine Learning in User Modeling 24

A Taxonomy of Modeling Methods 25

The Feed Reader Case Study 28

4 Exploring Modeling Methods 31

Content-based Methods 33

Collaborative Methods 41

Hybrid Methods 48

5 Concluding Thoughts 57

User Modeling on the Web 57

User Modeling in the Case Study 58

Future Work 59

Conclusions 61

References 62

Introduction

The World Wide Web presents vast opportunities for people to access its seemingly endless supply of information. This colossal repository of data in all shapes and sizes reaches new heights as more and more of the world's population gain access to its possibilities.

However, the Web's previously unimaginable amount of data presents new problems in need of novel solutions. One of these is the problem of *information overload*.

1.1 The Problem

Information overload occurs when an application presents more information than a user is able to consume. There are plenty of potential culprits: a poor signal to noise ratio, where relevant content is drowned out by useless information[1]; an interface ill suited to the current user's presentational needs or to the content being presented; a system that constrains the user, rather than adapt to how it is used.

Each culprit can degrade the user experience. Also called *analysis paralysis* and *information fatigue syndrome*, information overload is hardly a new phenomenon, with early research describing city dwellers experiencing an incapacitating overload of sensations in a new urban world (Edmunds and Morris, 2000).

The problem is especially pervasive on the Web. The sheer wealth and scope of information can quickly strain the user experience (Bawden and Robinson, 2009). Information published on websites can be ranked in a number of dimensions: Articles can be relevant or useless, unique or duplicate, urgent or outdated, structured or incoherent. Websites come with a certain authority, a number of followers and likely some detractors, all within widely differing structures, publishing schedules and agendas.

As we shall see, information overload is a huge challenge for *content aggregating websites*[2]. A content aggregating website is a system where content arrives from multiple sources of varying interest and relevance to each visiting user. Whether the system has multiple authors, pertain to multiple topics, or collate information from multiple other websites, the combined stream of information often result in a mismatch between canonical and individual priorities.

[1] In the *signal to noise ratio*, the signal is the content that is relevant to the user. The noise is any other content that is not relevant. When the ratio is low, useless content can prohibit proper information consumption.

[2] There are many examples of content aggregating websites. The New York Times (http://nyt.com) aggregates content from different sources, authors and topics. Google News (http://news.google.com) collates headlines from different websites. The social bookmarking service Delicious (http://delicious.com) lets people share bookmarks with other users.

1.2 Possible Solutions

The guiding assumption of this paper is that *personalization of content is the key to curbing information overload in content aggregating websites*. In other words, to improve the individual user experience, content should be prioritized, filtered and presented in a way that matches the preferences, interests and abilities of each user.

At first glance, personalization might seem like a trivial and obvious benefit. Yet, there are few extensive examples found in the wild. The most known applications of personalized content includes individual recommendations of products for sale, or individually targeted ads from centralized providers[3]. There seems to be few websites that take personalization to the next level, and leverage individual preferences throughout their content (see Figure 1.1). Why should the ads on a website be more adapted to the individual user than the content of the site itself?[4]

Figure 1.1: **Priority Inbox:** Google's Gmail provides an interesting approach to personalized content through their Priority Inbox email sorting system. Emails are sorted into groups based on how important the system believes they are to the user. Figure From Google. See http://mail.google.com/mail/help/priority-inbox.html

As one might gather from this lack of real world examples, personalization of online content is no simple matter. The act of individually adapting the content of a website raises a number of questions. First, there is the question of how to acquire precise knowledge about each user. Second, how will each user react to initiatives taken by the application? And as importantly, how can an interface best leverage what the system knows about its users? Before diving into these issues, let us start at the beginning.

The fields of Artificial Intelligence (AI) and Human-Computer Interaction (HCI) share a common goal of curbing information overload. However, as described by Lieberman (2009), their efforts are seldom combined: while AI researchers often view contributions from HCI as trivial cosmetics, the HCI camp tends to view AI as unreliable and unpredictable — surefire aspects of poor interaction design.

User modeling is one way of fighting information overload, where lessons from both fields are required — computational power and proper interface integration are equally important. User modeling methods strive to create predictive models of each individual user, either explicitly through feedback provided by the users, or implicitly through data mining and passive observation.

In AI, *user modeling* refers to precise algorithms and methods that infer knowledge about a user based on past interaction (Webb et al., 2001; Pazzani and Billsus, 2007; Schafer et al., 2007; Burke, 2007). By examining previous actions, predictions can be made of how the user will react to future information. This new knowledge is then embedded in a model of the user, which can predict future actions and reactions[5]. For instance, an individual user model may predict how inter-

[3] The Amazon web store relies heavily on personalized recommendations, suggesting new products to buy based on previous purchases. See http://amazon.com

[4] Facebook has through their social graph created a highly targeted advertising platform. Advertisers can select the exact attributes of who they think should see their ads. See http://facebook.com/advertising.

[5] For the purpose of personalizing information, *recommender systems*, a subset of the user modeling field, will be our focus. These systems allow for estimation of how relevant each item of unseen information is to each user. Recommender systems are introduced in Chapter 3

esting an unseen article will be to a user, based on previous feedback on similar articles or the feedback of similar users.

HCI aims to meet user demands for interaction. User modeling plays a crucial role in this task. Unlike the formal user modeling methods of AI, user models in HCI are often cognitive approximations, manually developed by researchers to describe different types of users (Fischer, 2001; Jameson, 2009; Cato, 2001). These models are then utilized by interaction designers to properly design the computer interface based on a models predictions of its user's preferences[6].

In earlier years, the field of user modeling was fragmented by its multidiscipline nature. However, as described by (Kobsa, 2001b), recent research has blurred the lines between the AI and HCI in user modeling.

[6] Totterdell and Rautenbach (1990) describes user modeling in interaction design as a collection of deferred parameters: "The designer defers some of the design parameters such that they can be selected or fixed by features of the environment at the time of interaction [...] Conventional systems are special cases of adaptive systems in which the parameters have been pre-set."

1.3 Objectives

The next chapter presents the problem of information overload, in detail. We present the methods of adaptive interfaces and personalized content as tools that help mitigate the problem.

We shall then look at how web content can be adapted on different levels of user granularity through methods from *user modeling* (Chapter 3). Specifically, we will look at the methods known as *recommender systems*. Our review presents a number of fundamental and novel approaches while keeping us on topic through an overarching taxonomy for user modeling methods (Chapter 4).

Finally, we will analyze what the different methods have in common, their possible uses, how they interact with an interface, and how the methods can be used in to mitigate information overload in a case study of an online feed reader (Chapter 5). Table 1.1 gives an outline of the upcoming chapters.

§	Title	Contents
2	Information Overload	Presents the problem in detail, and how adaptive content can help user overcome the problem.
3	User Modeling	An introduction to user modeling, recommender systems. Presents a taxonomy for modeling methods and an applicable case study.
4	Exploring Modeling Methods	Presents many different modeling methods in the context of the taxonomy and case study from §3.
5	Discussion & Conclusions	Discussions on common themes in user modeling, a presentation of possible future work, and finally, some concluding thoughts.

Table 1.1: **Chapter Outline**

Information Overload

In simple terms, *information overload* conveys the act of receiving *too much information*[1]. The problem is apparent in situations where decisional accuracy turns from improving with more information, to being hindered by too much irrelevant data (Eppler and Mengis, 2004) (see Figure 2.1).

The overload is often likened to a *paradox of choice*, as there may be no problem acquiring the relevant information, but rather identifying this information once acquired. As put by Edmunds and Morris (2000): "The paradox — a surfeit of information and a paucity of useful information."

While normal cases of such overload typically result in feelings of being overwhelmed and out of control, Bawden and Robinson (2009) points to studies linking extreme cases to various psychological conditions related to stressful situations, lost attention span, increased distractibility and general impatience.

Kirsh (2000) argues that "the psychological effort of making hard decisions about *pushed* information is the first cause of cognitive overload." According to Kirsh, there will never be a fully satisfiable solution to the problem of overabundant information, but that optimal environments can be designed to increase productivity and reduce the level of stress through careful consideration of the user's needs.

Economists provide a helpful perspective on information overload through the study of *attention economy*. In this context human attention is a scarce commodity, offset by how much irrelevant noise is present at any given time[2]. *Attention* can then be defined as "... focused mental engagement on a particular item of information. Items come into our awareness, we attend to a particular item, and then we decide whether to act" (Thomas H. Davenport, 2001).

E. Horvitz (2003) found that our limited attention pool should be a central consideration when designing new systems and interfaces: "Over the last five years, our team at Microsoft Research has explored, within the Attentional User Interface (AUI) project, opportunities for enhancing computing and communications systems by treating human attention as a central construct and organizing principle."

To evade information overload is then to maximize available attention, allowing more focus on the most important items of an interface.

[1] Information overload is a widespread phenomenon, with as many definitions as there are fields experiencing the problem. Examples include *sensory overload*, *cognitive overload* and *information anxiety* (Eppler and Mengis, 2004).

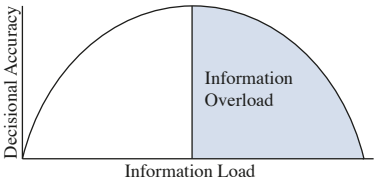


Figure 2.1: **Information Overload:** When additional data encumbers decisional accuracy. Figure adapted from Eppler and Mengis (2004)

[2] Herbert Simon was among the first to describe the concept of attention economy (Greenberger, 1971): "...in an information-rich world, the wealth of information means a dearth of something else: a scarcity of whatever it is that information consumes. What information consumes is rather obvious: it consumes the attention of its recipients. Hence a wealth of information creates a poverty of attention and a need to allocate that attention efficiently among the overabundance of information sources that might consume it."

2.1 Clues From Interaction Design

Conceptual models used in interaction design can help us see *when and where* information overload interferes with the user experience. Norman (1998) advocates a model called the *seven stages of action*, describing how each user goes through several states while using a system. First, the user forms a goal and an intention to act. The user then performs a sequence of actions on the world (the interface) meant to align the perceived world and the goals (see Figure 2.2). After performing a set of actions, the new world state is evaluated and perceived. At last, the user evaluates the perception and interpretation of the world in accordance with the original goal.

Information overload can interfere both before and after any action is taken. For example, if the application presents too much content, or presents content in a confusing manner, it can be difficult for the user to identify which actions that would help achieve the current goal. Likewise, after actions are taken, the new world state can suffer the same shortcomings of overwhelming scope or lack of presentations, leading to information overload. This precludes the user from properly evaluating the resulting application state. In short, an application interface can fail both before and after a user tries to interact with it.

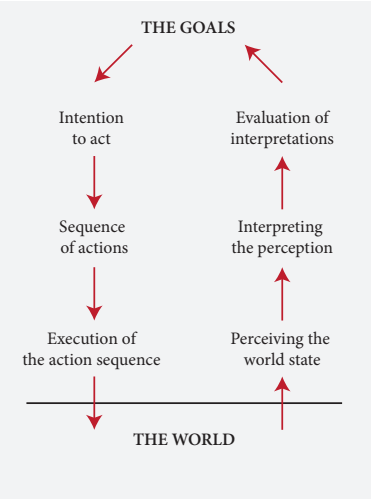


Figure 2.2: **The Seven Stages of Action:** This interaction design framework shows how information overload may occur in multiple phases of interaction. A user forms goals, performs actions and perceives the resulting system state. Information can prohibit the action forming phase, or the state interpretation phase. Figure adapted from Norman (1998).

Cato (2001) presents another conceptual interaction model that clarifies *when and where* information overload occurs. The Awareness, Understanding, Action (AUA) model says that the user has a certain information need, and seeks an information outcome. The purpose of using the system if to attain the needed information (See Figure 2.3).

To achieve said purpose, the user makes a number of choices based on what the current system presents, and performs a set of actions based on these choices. The system responds, and the user evaluates the resulting interface to form an understanding of the new information. The new understanding is the new user knowledge. User satisfaction is based on how this new user knowledge coincides with the desired information outcome.

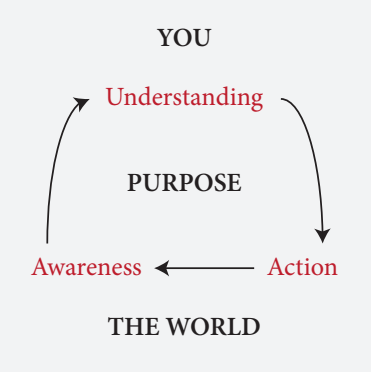


Figure 2.3: **Awareness, Understanding, Action:** The AUA model in simple terms. Through an understanding of their goal or purpose, a user forms actions to perform on the world (the system). Through an awareness of the resulting world state, user satisfaction is high if the awareness matches the current goals of the user. Figure adapted from Cato (2001).

As with the *seven stages of action* model, we see that information overload can cloud the users progress in two phases: Before performing actions and when receiving the result. If the application presents a lackluster interface when the user tries to find out what to do in order to achieve a goal, the right actions might not be taken. If the application presents the results of actions in a poor way, the new user knowledge will probably not match the desired information outcome. It is clear from these models that information overload happens throughout the interaction process, which is important to know when considering possible solutions.

2.2 Online Overload

There are multiple reasons why the World Wide Web incites information overload. As mentioned, the wealth and scope of data are natural culprits, as well as the varying dimensions of websites publishing the information. However, lessons from graph-theory can help us see why information overload occurs on the Web.

2.2.1 *Networks, Hubs & Continents*

Graph theory presents applicable models of the Web that characterize how people navigate between websites, and show how content aggregators form important hubs in the network. Here, nodes correspond to websites and directed edges between nodes are links from one page to another. The *degree* of a node is defined as its number of edges.

The Internet has the properties of a *small-world network* (Newman et al., 2000), a type of random graph, where most nodes are not neighbors, but most nodes are reachable through a small number of edges (See Figure 2.4). This is because of important random shortcuts differentiating the graph from a regular lattice. The graph is not random, but neither is it completely regular[3]. As described by Barabasi et al. (2003, p37), the average number of outbound links from a webpage is around 7. From the first page, we can reach 7 other pages. From the second, 49 documents can be reached. After 19 links have been traversed, about 10^{16} pages can be reached (which is more than the actual number of existing web pages, since loops will form in the graph).

The high degree of the Web graph would suggest that finding an optimal path to your desired page is quite difficult. Yet, while it is true that finding the *optimal path* is hard, finding *a good path* is not that big a challenge. When people browse the Web, links are not followed blindly — we use numerous different heuristics to evaluate each link, often resulting in a quite good path to where we want to go. So why is the Web still quite challenging to navigate?

As discovered by Albert and Jeong (1999), the Web also exhibits properties of a *Scale-Free Network* (SFN). They found that in some natural observed networks, there exists a small number of nodes with an extremely high degree. This is also true on the Web — some websites have a huge number of outbound links.

For comparison, while a random network is similar to a national highway system, with a regular number of links between major cities, scale-free networks are more like an air traffic system, with central hubs connecting many less active airports (Barabasi et al., 2003, p71).

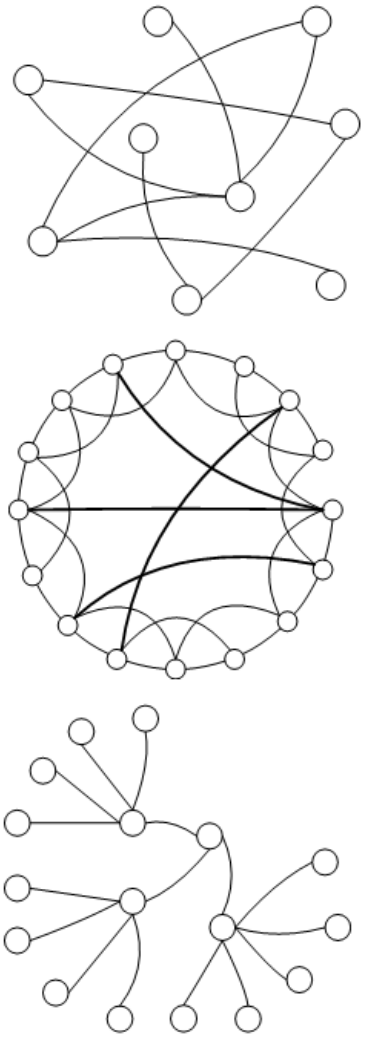


Figure 2.4: **Complex Networks:** Top down: A *random* network, a *small-world* network and a *scale-free* network (which is a type of small-world network). Figure from Huang et al. (2005).

[3] The concept of small-world networks came from the observation that there is often a surprisingly short minimum distance between nodes in an average graph. This is called the small world phenomenon, and is often mentioned alongside the concept of "six degrees of separation", popularized in a play by John Guare. This concept states that no person is on average more than six social links removed from any other person on earth.

These highly connected nodes, called *hubs*, are not found in small-world networks or random graphs. As demonstrated by the presence of hubs, the degree distribution of a scale-free network follows a power law, $P(k) \sim k^{-\gamma}$, where $P(k)$ is the probability of a node having k connections and γ is a constant dependent on the type of network, typically in the range $2 < \gamma < 3$. Since the Web has directed edges, we have Equations 2.1 & 2.2.

$$P_{in}(k) \sim k^{-\gamma_{in}} \tag{2.1}$$
$$P_{out}(k) \sim k^{-\gamma_{out}} \tag{2.2}$$

Albert and Barabási (2002) describes a number of studies placing the γ values for the Web in the $[2,3]$ range, with γ_{out} being slightly higher than γ_{in}. Both these probabilities exhibit power tails as seen in Figure 2.5. In other words, a few important nodes have a huge number of inbound and outbound links — the hubs. Barabasi et al. (2003, p86) proposed that hubs emerge in a scale-free networks because of two factors:

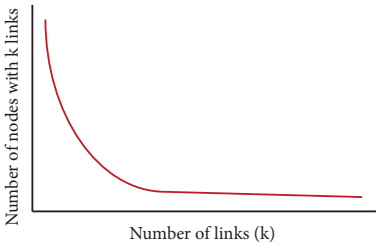


Figure 2.5: **sfn Long Tail:** In a scale-free network, most nodes have a relatively few number of links, while a few has a comparatively huge number of links. These nodes are called *hubs*, and their degree is described by a power law.

1. Growth: Nodes are added to the network one by one, for example when new websites are added to the Internet.

2. Preferential attachment: When new nodes are created, they connect to existing nodes. The probability that the new node will connect to an existing node is proportional to the number of links the existing node has. In other words, older, more established and central nodes are preferred neighbors.

This is called the Barabási-Albert model (Albert and Barabási, 2002), and the probability for a new node connecting to an existing node is given by $\prod k_i$ in Equation 2.3, where k_i is the number of links pointing to node i.

$$\prod(k_i) = \frac{k_i}{\sum_j^N k_j} \tag{2.3}$$

Another important topological observation of the Web is how fragmented it is. Barabasi et al. (2003, p166) describes the Internet as a number of continents:

- The first continent is the *central core*, where most important hubs reside. On this continent, most sites are easily reachable through each other.
- The second is the *in-continent*, and is defined by the group of websites that often link to sites in the central core, but seldom receive reciprocal links from central hubs.

- The third is the *out-continent*, which are comprised of sites that are often linked to from the central hub, but seldom link back.
- Finally, the Internet has many *islands*, or dark nets, which are not accessible through links from other continents[4].

[4] The islands of the Web is why search engines allows web masters to manually add their sites to the search engine index — not all sites are discoverable by following simple links. Some of the Web is topographically close, but a lot of it is not.

Returning to our main topic, the *hubs* often represent the previously mentioned *content aggregating websites*. Search engines, social link aggregators, news portals, et cetera are all hubs of the Internet, emerging from the preferential link attachment of newly created nodes.

Factor in the existence of multiple sub-graphs, or continents, and we can intuitively see that navigating the Web is not as easy as it might appear from simple models.

What does seem clear is that these content aggregating hubs are prime candidates for overwhelming their users with information. The fundamental observed structure of the Web creates the need for information brokers that link the net together, and the need for techniques to display a lot of data.

2.3 Adaptive Content

How do we approach solving the information overload problem on the web? Our approach is to adapt the application content to the preferences and properties of the current user. As mentioned, the assumption is that *personalization of content is the key to curbing information overload in content aggregating websites*. Let us therefore examine the properties of an interface that adapts to each user.

2.3.1 *Interface Autonomy*

An autonomous interface is one that takes initiatives on its own, regardless of whether the user has asked for it. Creating adaptive interfaces means increasing the level of autonomy in an interface, a technique that raises many interesting issues. Naturally, any application that automatically personalizes its content will be autonomous to some degree.

Adaptive interfaces can be classified into increasing order of autonomy. At the order of least autonomous systems, we have *customizable interfaces*. These are interfaces that the user may customize themselves, but that do not take the initiative or change anything without explicit user action. For example, an interface might have a settings panel where users can change the order of items in a menu.

At the next level of autonomy, we have *adaptive interfaces* that suggest to the user possible changes or actions that might be beneficial. For example, an email application could suggest which folder an email should be moved to.

Figure 2.6: **Interface Autonomy:** Interfaces range from those only customizable by the user, to intelligent systems takes the initiative on its own accord.

At the most autonomous level, *intelligent interfaces* implicitly and automatically customize the interface or content based on passive observation of the user. This could for instance entail automatic filing of emails based on content classification and data mining of previous user actions with similar messages.

An application that personalizes content automatically will fall somewhere in the two last categories and present either an adaptive or intelligent interface, depending on the extent and transparency of its autonomy. As we wish to keep in mind the HCI perspective when performing user modeling, let us briefly consider the possible usability issues of personalized content.

2.3.2 *The Usability of Autonomy*

The field of HCI aims to create highly usable application interfaces. Shneiderman (1997) defines the core principles of creating usable designs as the following:

- The system should be intuitive and easy to use.
- The system should be effective once learned.
- The system's interface should be easy to remember.
- The system should be subjectively pleasant to use.

In addition, Cato (2001) suggests the following attributes of good usability that we will consider:

- Users should feel in control, not controlled.
- Users should feel supported and respected by the system.
- Users feel their privacy is well kept by the system.

From this, it is clear that a usable interface is rooted in predictability, effectiveness and intuitiveness. Extending our discussion to autonomous adaptive systems, a few challenges become apparent: An intelligent interface by definition changes the interface in ways not explicitly requested by the user. Considering the usability attribute of intuitiveness and ease of use, it is clear that the changes made by any adapting agent should not compromise expectations each user might have. In other words, avoid large and surprising changes.

Considering that the user should feel in control, and not feel controlled by the system, transparency becomes a crucial matter. Any adaption initiative taken by the system should

be overt, undoable and have a clear purpose. By creating a system capable of explaining and highlighting adaptions made, the control remains with the user.

Privacy is another important factor that intersects HCI and AI. While the user might want a system to perfectly adapt to his or her needs, the fact that a computer system knows a lot about their person might cause concern. As put by Cato (2001), a user needs to feel comfortable that the system helps protect information belonging to them or their clients. As the core principle of user modeling methods in AI is gathering, mining and synthesizing information about a person, balancing the competence of the model with a user's expectations of privacy is no easy matter.

Clearly, to create a successful adaptive interface, finding the right AI algorithm for inferring knowledge about a user is not enough. To create a truly usable system, the cited attributes of usability must be taken into account as well.

2.4 Solving the Problem

What have we found so far? First and foremost, that information overload is a big problem, especially on the web. Our goal then, should be providing users with interfaces that autonomously sorts, filters and presents its content in a helpful way, mitigating the overload problem.

Spam-filters deliver some of these features. However, our task is much more complex than removing information that none of the users wants. In the world of information overload, all available incoming information may be relevant and interesting, and to make matters worse, what is relevant and interesting is of course completely dependent on the individual user in question.

It seems we have enough ground to explore how the content of online aggregators can be adapted to each user. To do this, we will review the field of user modeling, where different approaches are used to estimate how relevant and interesting each information item personally is to each user.

More specifically, we will consider *recommender systems*, a subset of user modeling methods, which perform the exact functions we need to sort, filter and present content in a way that mitigates information overload.

User Modeling

The term *user modeling* (UM) lacks a strict definition. Broadly speaking, when an application is adapted in some way based on what the system knows about its users, we have user modeling. From predictive modeling methods in machine learning and how to implement these methods, to how interface design is influenced by personalization — the field covers a lot of ground.

We approach user modeling from the AI perspective. In this context, user modeling methods try to create digital representations of users, and then calibrate and adapt the interface or the content based on these models.

It is important to differentiate between adapting the interface of an application and the content of an application. Many user modeling methods strive to personalize the interface itself, e.g. menus, buttons and layout of interface control elements (Jameson, 2009; Fischer, 2001). Adapting the application content, on the other hand, means changing how and what content is displayed[1].

We are interested in adapting the content of an application since the source of our overload problem often comes down to a mismatch between presented content and desired content.

[1] For instance, interface adaption might mean changing the order of items in a menu, while content adaption might mean changing the order and emphasis of results in a web search interface.

3.1 Recommender Systems

We will focus on one type of UM methods that are used to personalize the content of an application: *recommender systems* (Adomavicius and Tuzhilin, 2005). A recommender system takes a set of items, and rates each item based on some metric, in the context of a single or group of users. Amongst other things, these ratings can then be used to recommend new items, filter out low ranked items and properly sort a list of many items[2].

Recommender systems provide many methods for combatting information overload. We can for example sort items in order of relevance, use the rankings to remove the least desired items, or emphasize new and interesting content. Let us take a look at what defines a recommender system.

In the formal description of a recommender system presented by Adomavicius and Tuzhilin (2005), C is the set of all users and S is the set of all items that can be recommended. The space of both sets C and S might be very large — consider extensive product recommendation systems or websites

[2] Most recommender systems are centralized and deeply connected to one system's users and items. However, alternatives exist. Ziegler (2005b) presents the emergence of decentralized, distributed infrastructures for recommender systems. Directed Edge is a company going in the opposite direction, building a centralized recommender graph which other sites can hook into. See http://directededge.com/.

with a huge number of users. Each user c and item s can be described by their own characteristics, e.g. the personal attributes of an individual and terms describing a product. The utility function u measures the utility (i.e. the rating) of an item s to user c:

$$u : C \times S \to R \tag{3.1}$$

Here, R is a totally ordered set of of items, ranked by their utility to each user. To create a recommendation for each user $c \in C$, we wish to select an item $s' \in S$ to maximize the utility. In other terms:

$$\forall c \in C,\ s'_c = \arg\max_{s \in S} u(c, s) \tag{3.2}$$

The utility function u depends on the modeling method being used, the active user and the item in question. The value of u can for instance be a user's explicitly submitted rating of a product.

The *reason* for using a recommender system is that the utility u is not defined for the entire $C \times S$ space, i.e. the system does not explicitly know the utility of each item for each user. The point of a recommender system is then to extrapolate u to cover the entire user-item space.

In other words, to be able to rank items according to user preferences, the system must be able to predict each user's reaction to items they have not yet explicitly rated themselves. This is where predictive user models come in handy.

3.2 Estimating Ratings

The input to a recommender system is usually a set of users C, a set of items S, and a sparse matrix of utilities linking specific users and items. To predict how much an individual will rate a specific unseen item, we need a method for estimating this utility, based on limited data.

Recommender systems fall into one of two categories based on how they perform this estimation. First we have the *model-based* approach, where the recommender system builds predictive models based on the known data. Unseen items can then be fed into this model to compute its estimated utility score. For example, creating a Bayesian networks from past interaction is a model-based approach.

The other category is the *heuristic* or *memory-based* approach. These methods use the raw data of items, users and ratings to directly estimate unknown utility values. For example, recommending items similar to the ones already rated

by computing the cosine similarity of their feature vectors is a heuristic approach.

In addition to the *model-based* and *heuristic* approaches, there is the orthogonal question of how data from the set of users, items and ratings should be used. This consideration puts recommender systems into three main categories: Content-based, collaborative and hybrid recommendations.

3.2.1 *Content-based recommendation*

Content-based recommendations regard an individual user's past history as predictive of future actions (Pazzani and Billsus, 2007). By only considering the individual user in adapting an application, highly tailored models can be created. However, such methods often require a lot of interaction before reliable models can be created (Adomavicius and Tuzhilin, 2005)[3].

When using content-based learning, the utility function $u(c, s)$ of user c and item s is extrapolated from $u(c, s_i)$, where $s_i \in S$ is an item similar to s.

[3] The problem of having to do complex inference from little data, as is often is in content-based learning, is often called the *sparsity problem* or the *cold start* problem. This is closely related to the problem of *overfitting* data, where the algorithms creates models that match the training data, but not the actual underlying relationships. A lot of research looks at ways to overcome sparse data, i.e. achieving "warmer" cold start.

3.2.2 *Collaborative recommendation*

Collaborative or social recommendations build predictive models for users based on the actions of similar users (Schafer et al., 2007). The observation is that similar users should have similar usage and action patterns. By using data from more than one user, expansive models may be built. These methods are especially useful when considering new users of a service. A central problem with collaborative methods is that the resulting model is not as individually tailored as one created through content-based learning. Collaborative models must be careful not to represent the *average* user, but a single individual.

When using collaborative learning, the utility function $u(c, s)$ of user c and item s is extrapolated from $u(c_i, s)$ where $c_i \in C$ is a user similar to c.

3.2.3 *Hybrid recommendation*

Because of *the new user problem* of content-based learning and the *average user problem* of collaborative learning, many systems use a hybrid approach (Burke, 2007). By combining content-based and collaborative learning, systems that properly handle predictions for new users and avoid too much generalization in the models can be achieved. Hybrid recommenders can be created in many ways, for instance by combining separate and weighted recommenders.

3.3 Machine Learning in User Modeling

User modeling relies heavily on methods from the field of Machine Learning (ML) (Zukerman and Albrecht, 2001; Webb et al., 2001; Adomavicius and Tuzhilin, 2005).

One of the simplest ways of extrapolating unknown utility values in the $C \times S$ space, is by using linear models. Such models combine weighted scalars to produce some descriptive value of the user in question. For example, if the task is to estimate how much user c_x likes document d_y, i.e. estimating $u(c_x, d_y)$, the result can be found by combining explicit ratings from other users, weighted by how similar those users are to user c_x[4].

Markov models are another ML method applicable to predictive user modeling, along with neural networks, bayesian networks, and rule induction (Zukerman and Albrecht, 2001).

Methods from ML are often focused on sets of documents, which makes them highly usable in adapting the content of a web application to a user. As our venue is the web, a document can be a website, an article on a website or something even more specific like the title or abstract of an article. When performing user modeling, being able to model the content to adapt is of course as important as modeling each user.

In addition to predictive modeling, document collection modeling is another field of ML important to UM. From TF-IDF calculation for comparing document similarity, to classifications and clustering algorithms for organizing a set of documents — performing user modeling requires methods from machine learning.

Because of the reliance of UM on ML methods, many of the challenges faced in ML must be considered when creating recommender systems. A few of these challenges, as described by Webb et al. (2001), are summarized in Table 3.1.

[4] The Google Prediction API is an interesting 3rd party recommender system. This HTTP REST-based API lets users submit datasets and get recommendations from Google's machine learning methods in return. See code.google.com/apis/predict

Challenge	Description
The need for large datasets	Many ML methods need a lot of data or many documents to produce reliable predictions. This is especially problematic in content-based learning, where predictions are based on a single user — a lot of interaction is required before any predictions can be made.
The need for labeled data	To create a predictive model, labeled training data must be provided to properly calibrate the model. This is especially problematic on the web, where documents come in any number of forms and formats, making reliable labeling a difficult task.
Concept drift	When a user's attributes, preferences or interests change over time, we have what is called concept drift. The model created by an ML method must be able to compensate for such drift by continuously refining and updating the model.
Computational complexity	Many ML methods suffers from a high computational complexity. When considering the size of the Internet and the numbers of visitors a popular website may have, this becomes a fundamental prohibiting factor for using many methods from ML in UM.

Table 3.1: **Fundamental challenges in machine learning**

3.3.1 *An Abundance of Choice*

As we have seen in this and the previous chapter, considering, evaluating and implementing a user modeling system is no simple task.

First, a developer has to consider whether personalization of content would benefit the user of the application in question — a question without a simple yes or no answer.

Second, personalization of content can mean so many things. Is the goal to remove irrelevant content? To emphasize the most interesting information? To recommend new, unknown content? The possibilities are many.

Third, as the purpose of personalizing an application comes to light, we have the question of how this should be done. If we want a recommender system, which predictive modeling method should be used? Which underlying ML method supports the prediction?

Finally, the task of connecting the personalization with the interface is no easy task. Some methods require explicit input from the user, which the interface must take into account. The other question is how information gained from modeling should be leveraged in the interface.

The next chapter will present a wide variety of different user modeling methods, that all make varying choices in machine learning method, knowledge acquisition and interface integration. However, first we need a proper framework to put the different modeling methods in the same context.

3.4 A Taxonomy of Modeling Methods

To have any chance of discussing and comparing different approaches in user modeling (recommender systems, to be precise) an applicable taxonomy will be helpful. First, we need a notion of the different dimensions in which each method should be evaluated (Table 3.3), and a simple taxonomy to succinctly compare the different methods (Table 3.2)[5].

In her seminal paper on user modeling, Rich (1979) outline three major dimensions for classifying user models: Do the models represent individual users, or a canonical user? Are the models constructed explicitly by the user, or implicitly extracted from data mining and passive observations? Do the models contain specific short-term information or long-term general information? As Rich describes, personalized models are only worth pursuing in systems with a highly heterogenous set of users. If they are indistinguishable, building a model of the canonical user is much more effective.

Uwe Malinowski and Schneider-Hufschimdt (1992) presents a number of descriptive dimensions for user modeling. We use a subset of this framework when describing modeling

[5] While the explicit performance of recommender system would be a natural metric in a survey of possible methods, we will not focus on performance beyond reporting the numbers cited in each paper. As this will be an exploratory survey, independent performance measures is beyond the scope of this paper. However, see Herlocker et al. (2004) for an introduction on how to evaluate collaborative recommenders.

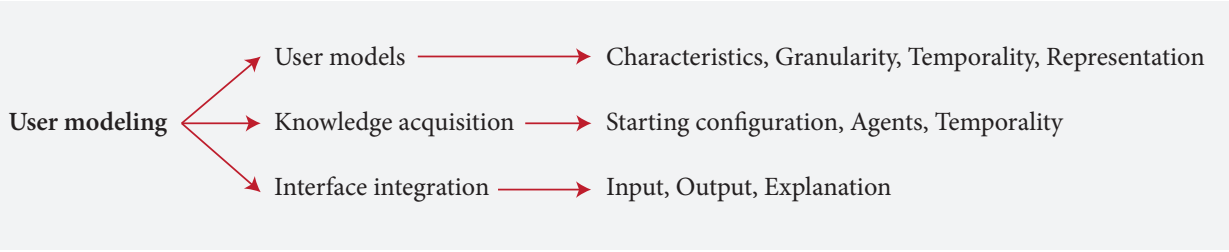


Figure 3.1: **Dimensions in user modeling:** A user modeling method can be classified by its model structure, the knowledge acquisition methods and the interface integration required to use the method.

methods, while also taking into account how the final interface supports and leverages the modeling method.

In this terminology, a user modeling method u is a triple comprised of a user model structure m, a knowledge acquisition method k and a method of interface integration i: $(m, k, i) \rightarrow U$. (See Figure 3.1). The elements of the triple are described by the aspects given in Table 3.3, which will guide our discussion of modeling methods.

However, to be able to succinctly compare different methods, we also need a much more precise taxonomy with fewer free variables. In that regard, each modeling method will be compared according to the variables in Table 3.2.

Variable	Values
Predictions	Content-based | Collaborative | Hybrid
Method	Heuristic | Model-based
Granularity	Canonical | Typical | Individual
Temporality	Short-term | Long-term
Agents	Implicit | Explicit

Table 3.2: **User modeling taxonomy:** Each variable has one value for each user modeling method. This will allow simple comparison of the most important aspects of each approach.

In this simple taxonomy, the *predictions* variable refers to how the approach performs the utility estimation in the $C \times S$ space.

The *method* variable says what general technique the approach uses to predict ratings (Adomavicius and Tuzhilin, 2005). As mentioned, *heuristic* approaches uses the raw user and item data to provide estimations. On the other hand, *model-based* approaches build explicit models from the data to provide indirect estimations.

The *granularity* variable tells whether this approach creates models for the canonical user, stereotypical users or individual users. *Temporality* refers to how volatile the gathered knowledge will be. The *agents* variable signifies whether the knowledge gathering and presentation is implicit and opaque, or explicit and requires dedicated user interaction.

Models	*Characteristics*	Which aspects of the user does the model represent? A model can synthesize many types of personal information, from preferences and interests, to experience and abilities.
	Granularity	How specific is the user model? The model can correspond to a canonical user, stereotypical users, by organizing users in descriptive groups, or individual users, by creating specific personal models.
	Temporality	How permanent is the knowledge stored by the model? Its content can be short-term, for example the last actions taken by a user, or long-term, for example representing seldom changing interests and preferences of a user.
	Representation	Is the knowledge represented by the model quantitative or qualitative? Quantitative information can for example be the user's click-through rate, number of visits or error rate. Qualitative information can be symbolic representations of interests, placement in stereotypes et cetera.
Knowledge	*Starting conf.*	Does the model start out with typical knowledge of a user, or does each modeling process begin with a blank slate? If the starting point is not a blank slate, where does the initial information come from?
	Agents	Does the user specify personal knowledge through manual input, or point to personal information sources, or does the system implicitly mine information from observation of interaction?
	Temporality	When does the knowledge acquisition occur? When systems rely on the user as the primary knowledge gathering agent, the information can be collated before any use of the system takes place. Conversely, the acquisition can be a continuous task extending across each user session.
Interface	*Input*	How does the user modeling method rely on input from the interface? If the gathering method is explicit, interface elements which allows feedback from the user is required.
	Output	How is the user model utilized to adapt the user interface? Many methods reliably describe how a user model is created, but seldom explain how this knowledge can best be used to enhance the final interface.
	Explanation	Does the user modeling method explain its decisions to the user? If the interface adaption is subtle, or requires acceptance or rejection from the user, the interface must allow for appropriate explanation.

Table 3.3: **Descriptive dimensions of user modeling methods**

3.5 The Feed Reader Case Study

To explore different methods of user modeling, a helpful case study will be used throughout the discussion. Our case is an *online feed reader*, called Signal, which we will use to explore how each method could be used in a real world example of a content aggregating web application[6].

[6] Naturally, the exact specifics of our case study is not fundamentally important to any method described. Any website clamoring to rein in a continuous stream of documents for different users could benefit from the visited topics.

3.5.1 *A Feed Reader*

We define an online feed reader as any web application that allows users to subscribe to a set of websites. The feed reader monitors the sites and presents the user with a collated stream of new articles from his or her sources. This allows each user to stay up to date on their topics of interest, without having to manually visit a number of websites.

In a feed reader, a *source* is any website that makes its content available through the use of feeds, for example by using the RSS or Atom (Nottingham and Sayre, 2005) protocols. A feed is a machine-readable XML file that documents the latest content published by the website. When a user adds a new website to their personal list of sources, the feed reading application begins monitoring (Gregorio and de Hora, 2007) the XML file for changes (see Listing 3.1). When new content is published, the application downloads the new information (often just a summary of the new content), and shows the news to each user subscribed to this website.

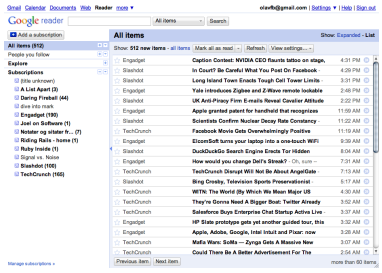


Figure 3.2: **Online Feed Reader:** Google Reader is a contemporary online feed reader. The user's source websites are shown on the left. On the right is a combined chronological list view of articles published by the sources. Image © Google 2010. See http://reader.google.com

Listing 3.1: **Atom feed**: A plain Atom XML file with one content entry.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Example Feed</title>
  <link href="http://example.org/"/>
  <updated>2003-12-13T18:30:02Z</updated>
  <author><name>John Doe</name></author>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>

  <entry>
    <title>Atom-Powered Robots Run Amok</title>
    <link href="http://example.org/2003/12/13/atom03"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2003-12-13T18:30:02Z</updated>
    <summary>Some text.</summary>
  </entry>
</feed>
```

Figure 3.3: **Native Feed Reader:** NetNewsWire, a popular native feed reader for MAC OS X, shows the same general interface structure as Google Reader. Sources are listed on the right, while items from each source are on the left. Image © NewsGator ® Technologies, Inc 2010. See http://netnewswireapp.com/mac/

Feed readers, both online and as native applications, present notoriously homogenous interfaces (see Figures 3.2, 3.3, 3.5 & 3.4). The universal style for such applications seem to be a flat or hierarchical list of websites on one side, and a list of new content on the other side, with titles and additional excerpts.

3.5.2 *Information Overload in Feed Readers*

Feed readers often inflict information overload on their users. The combined list of articles is often presented verbatim, in a chronological list, without regard to user preferences or relations between content items. While the initial goal of *push* technologies such as feed readers would seem to be stopping information overload, the resulting torrent of new information can quickly turn *push* to *shove*, as described by Edmunds and Morris (2000). When a user already suffers from information overload, a service that appears to push new information to them may seem as a new annoyance, providing little value.

Without modeling, users may have to scour through multiple articles on the same topic spread amongst other topics. Without modeling, the feed reader will continuously present the user with irrelevant articles if published by the sources. And without a helpful interface assisting the user in navigating and understanding the never ending torrent of data, information overload remains.

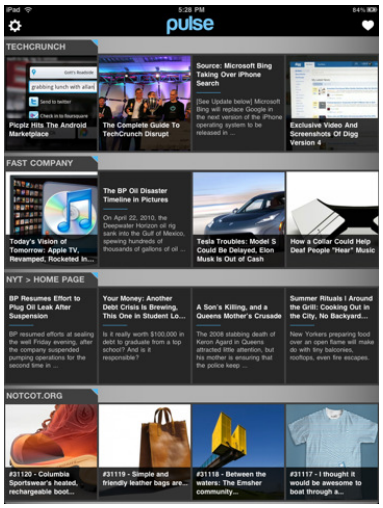


Figure 3.4: **Tablet Feed Reader:** Pulse is an application the Apple iPad, which sports an interface quite different from traditional readers. The content is aesthetically enhanced by focusing on images and a clever layout. Image © Alphonso Labs Inc 2010. See http://itunes.apple.com/app/pulse-news-reader/id371088673?mt=8

3.5.3 *Our Application: Signal*

Our application will be a simple feed reader with the following characteristics:

- The application has many users, and each user has a personal set of subscribed websites.
- We are only interested in how to best present new content from all the user's sources, not from one single source.
- Our goal is to adapt the list of new content to the preferences, interests and reading habits of the user.

It is important to note our assumption that a user does not always want all articles from all their sources. If every element was as important, changing their presentation through user modeling would have no value. We therefore assume that for each user, some of the sources are more important than others (e.g. more interesting websites), and that content produced by a single website varies in importance to the user (e.g. a user might not be interested in every topic a website concerns itself with).

As such, Signal, our application, will be a useful tool for exploring different AI and interface design methods that may contribute to its efforts to adapt its content to the current user.

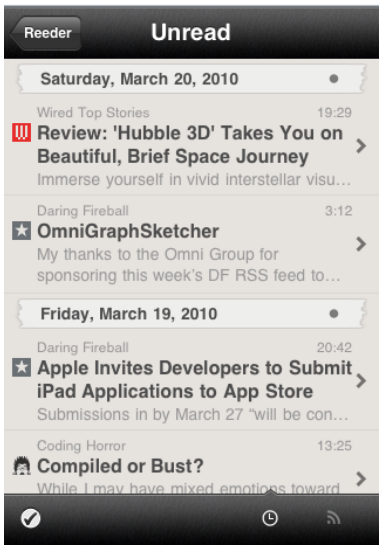


Figure 3.5: **Mobile Feed Reader:** Reeder is a feed reader for the Apple iPhone, with a compact list of sources and news optimized for a small screen. Here, new items from the user's sources are listed chronologically. Image © Silvio Rizzi 2009. See http://reederapp.com/2/

Exploring Modeling Methods

As one might conclude from the many different dimensions of user modeling, the literature describes a staggering number of different techniques. This chapter will select a few of the most prolific, seminal as well as novel methods, to give an overview of where the field of user modeling through recommender system stands today.

The papers described in the following sections were selected based on two criteria: First, four existing literature reviews from the field were collected. These can be seen in Table 4.1. Each of the reviews were selected based on how many times each have been cited in other papers, and based on how well they compliment each other.

Adomavicius and Tuzhilin (2005) gives a thorough review of state-of-the-art methods in recommender systems, in a number of dimensions. Pazzani and Billsus (2007), Schafer et al. (2007) and Burke (2002) are more focused and present approaches within content-based, collaborative and hybrid recommendations, respectively. Articles from each review were selected based on how well they represent each approach, complement each other, and whether they were found in more than one review.

In addition to articles found in each review, other articles were selected based on the novelty of their approach, and their respective number of citations. A summary is given in Table 4.3. Each row gives a paper's author, a short summary of their approach, and an evaluation of the presented method to put them in the same context. The evaluation of each method follows the taxonomy from in Section 3.4. Explanations for the column keys is given in Table 4.2.

#	**Literature reviews & Introductions**	
1	Adomavicius and Tuzhilin (2005)	Toward the Next Generation of Recommender Systems
2	Pazzani and Billsus (2007)	Content-based recommendation systems
3	Schafer et al. (2007)	Collaborative filtering recommender systems
4	Burke (2002)	Hybrid Recommender Systems: Survey and Experiments

Table 4.1: **Literature Reviews**

Keys and values for the modeling techniques table				
M (method)	M	Model-based approach	H	Heuristic approach
G (granularity)	I	Individual models	T	Typical, group based models
T (temporality)	S	Short-term knowledge	L	Long-term knowledge
A (agents)	I	Implicit knowledge gathering	E	Explicit knowledge gathering

Table 4.2: **Keys for Table 4.3**

Content-based modeling methods	M	G	T	A	#
Balabanović and Shoham (1997): Distributed knowledge agents deliver recommendations	M	T	L	E	1,2,3
E. Horvitz (2003): Bayesian network from physical and digital user attention metrics	M	I	S	I	
Lang (1995): Treats ratings as categories and rate unseen articles by IR classification methods	M	I	L	E	1,4
Magnini and Strapparava (2001): Individual synset graphs from semantic modeling (§4.1.3)	M	I	L	I	
Pazzani and Billsus (1997): Bayesian classifier used to incrementally learn user models	M	I	L	E	1,2
Rich (1979): Hierarchical stereotypes inferred by natural-language processing (§4.1.1)	M	T	L	E	1,4
Smyth (2007): Using case-based reasoning for structured recommendations (§4.1.2)	H	I	L	E	3
Collaborative modeling methods					
Billsus and Pazzani (1998): Reducing the user-item dimensionality with SVD (§4.2.2)	M	I	L	E	1,3
Konstas et al. (2009): Social graph traversal with random walks and restarts (§4.2.1)	H	I	L	E	
Resnick et al. (1994): Rating servers recommend items based on ratings of similar users	H	I	L	E	1,3,4
Shardanand and Maes (1995): Explores different algorithms for computing user similarity	H	I	L	E	1,3,4
Ujjin and Bentley (2002): Fine-tuning user profile matching with a genetic algorithm (§4.2.3)	M	I	L	E	
Walter et al. (2008): Collaborative filtering through a social network of transitive trust	H	I	L	I	
Hybrid modeling methods					
Al-Shamri and Bharadwaj (2008): Hybrid rating/content user models with fuzzy features	M	I	L	E	
Basu et al. (1998): Combining item content and ratings into descriptive hybrid features	M	T	L	E	1,3,4
Claypool et al. (1999): Weighted average of content-based and collaborative methods (§4.3.1)	H	I	L	E	1,3,4
Debnath et al. (2008): Collaborative item feature weighting through social network analysis	H	I	L	E	
Hotho et al. (2006): Vector space model with folksonomy-based similarity measures (§4.3.3)	H	I	L	E	
Huang et al. (2002): Graph traversal through a shared user-item graph (§4.3.2)	H	I	L	I	
Ziegler (2005a): Distributed recommender system using the semantic web	M	I	L	E	

Table 4.3: **Selected papers describing user modeling systems:** Explanation for each column is given in Table 4.2. The last column (#) shows in which of the literature reviews from Table 4.1 the paper can be found. See Chapter 5 for an analysis of this table.

Following the lead of the literature reviews from Table 4.1, the different approaches fit nicely into three main categories: content-based and collaborative recommendation systems, and hybrid systems that combine the two. Note that the placement of a paper in one category does not mean it is entirely unsuitable for one of the other two approaches — each technique often employs a host of different strategies. However, for clarity, each method is placed according to its overarching and most prolific approach.

While each approach given in Table 4.3 is placed according to our taxonomy, a much more detailed explanation is needed to appreciate the sheer complexity of recommender systems.

To convey these dimensions, the rest of the sections in this chapter presents a subset of the selected papers, in detail. We will consider each method in light of the different dimensions of user modeling given in Table 3.3[1].

Each category of methods will also be evaluated by their application to our case study, as presented in Section 3.5. By examining how content-based, collaborative and hybrid approaches can help improve an online feed reader, we will get an idea of the actual value provided to the end user.

Chapter 5 will summarize and analyze the results of our taxonomic placement of the selected recommender systems. For now, let us take a look at nine exciting modeling methods, starting with a seminal paper from 1979 on stereotypes.

[1] These dimensions include how the user models are created, what they represent, how and when knowledge is gathered, and how the method can be integrated with an interface. In short, there is a lot of aspects to consider when evaluating a user modeling method.

4.1 Content-based Methods

This section presents three methods that rely on content-based rating estimations, the simplest form of recommender systems. Each method matches users and items based on stored knowledge of an individual user and knowledge of the items to recommend.

We will also look at the different general dimensions of content-based modeling, and how these methods can be integrated with the interface of our case study application.

Modeling methods in this section	**M**	**G**	**T**	**A**	**#**
Magnini and Strapparava (2001): Individual synset graphs from semantic modeling (§4.1.3)	M	I	L	I	
Rich (1979): Hierarchical stereotypes inferred by natural-language processing (§4.1.1)	M	T	L	E	1,4
Smyth (2007): Using case-based reasoning for structured recommendations (§4.1.2)	H	I	L	E	3

Table 4.4: **Content-based methods**

4.1.1 *Hierarchic Stereotypes*

The first modeling method employs the *stereotype* concept to model user characteristics. Rich (1979) presents a system designed to recommend books from a library to a user. Knowledge of the individual is gained through a dialogue between the user and the system, which result in one or more stereotypes being added to the user synopsis (USS). This is a content-based, individual approach for recommending items based on user traits.

A stereotype is a collection of *facets*. Each facet is a characteristic or attribute of users belonging to the stereotype. Inside a stereotype, every facet has a *value* which gives the relational strength between the type and the characteristic. Note that the value can be negative, to retain knowledge of inverse characteristics.

A facet is also imbued with a rating, which signifies the probability that a person in this stereotype is accurately described by the rated facet. Table 4.5 is an example of a "sports-person" stereotype. In this example, facets have values from -5 to 5 and ratings range from 0 to 1000, where a higher rating indicates a greater probability.

Facet	Value	Rating
Interests-sports	4	800
Thrill	5	700
Tolerate-violence	4	600
Romance	-5	500
Education	-2	500
Tolerate-suffering	4	600
Physical-strength	4	900
Perseverance	3	600
Activated-by	Athletic	
Generalization-of	Any-person	

Table 4.5: **Sports-Person Stereotype**: A set of facets describing a person in the sports-person stereotype, the types generalization and the trigger for inferring membership.

The *generalization* part of the stereotype refers to a hierarchical organization of all types. A directed acyclic graph (DAG) is used to represent relationships between types, where a descendant of a type means that this type is a specialization of its parent. For instance, in the sports-person example, the type os a specialization of the *any-person* type, which is the root of the DAG.

The *any-person* root stereotype provides default values for all facets, but with very low ratings. As Rich (1979) puts it: "These values can be used to prevent the system from doing

anything that might be offensive until it learns enough about the individual user to know his particular inclinations. It is a cross between a model of a canonical user and a lower bound on the user's tolerance for things."

To infer a users membership to a particular stereotype, the types also have mechanisms called *triggers*. A *trigger* is an event or situation which, upon activation, places the stereotype in the user synopsis (USS) (see Table 4.6). This is the model of the user which contains information on each stereotype the user is a member of. Each trigger imbues a rating, describing how likely its activation is to mean the user belongs to its activated stereotype. In the sports-person example, the trigger is *athletic*. In Rich's paper, triggers are activated through a dialog between the user and the interface, using key-words to infer activation of triggers.

Facet	VR	Reason
Gender	F, 1000	Name
Nationality	USA, 100	Any
Education	5, 900	Intellect
Seriousness	5, 800	Intellect

Table 4.6: **User Synopsis:** (VR = Value and Rating) A small part of a USS, where the user is modeled by combining different facets from each stereotype the user is inferred to be a member of. Note the *nationality* facet, which inherits its value from the general stereotype, with a very low rating.

Dimensions The USS models produced by the stereotype captures characteristics of users in predefined atomic categories. Based on a dialogue between the user and the interface, triggers are activated to place users in stereotypes. Any characteristic about a user can in theory be represented by the system, but, at least in this example, manual input is required.

The granularity of the hierarchical stereotype models lie between the canonical and individual user. The number of stereotypes, and how many facets are required to form one decides just how individual the final user model becomes. If the stereotypes have few facets each, the union of facets in all a users stereotypes can produce quite a personalized model.

Temporally, the knowledge retained is of a long-term characteristic. As this example deals with modeling user interests and attributes, the knowledge found in the models should remain relevant for a long time.

Variable	Value
Predictions	Content-based
Method	Model-based
Granularity	Typical
Temporality	Long-term
Agents	Explicit

Table 4.7: Taxonomic placement of hierarchic stereotypes.

The starting configuration for new users of the system is a membership in the *any-type* root stereotype, which is akin to calling it a blank slate, given the low ratings in this type. However, as the initial use of the system is a written dialogue with the user, memberships in additional stereotypes are quickly inferred before the system uses the resulting model.

The knowledge gathering agent in this system is the dialogue engine, with its natural language capabilities and use of key-words to place users in stereotypes.

Temporally, the knowledge is gathered throughout the process. As the dialogue progresses, the system will soon produce a book recommendation that the user may accept or reject. If the suggestion is rejected, the dialogue and gathering process continues.

In our example, the only interface integration is through the written dialogue. Input is written by the user and output

displayed in the same form. The system is also capable of explaining why certain books were suggested by telling the user which stereotypes they have become part of, and receiving feedback about these memberships.

The taxonomic placement of this recommendation strategy is given in Table 4.7.

Stereotypes in Signal It should be clear that the main strength of user modeling through stereotypes is providing recommendations. By placing both the users and the content into stereotypes, new content in a stereotype can be recommended to a member of the same type.

In a feed reader such as Signal, stereotypes could be used to point out new sites a user should subscribe to. If a website produces content that intersect with some percentage of the user's type memberships, recommending this site to that user seems like an educated suggestion.

So how do we place users in stereotypes? By labeling and performing keyword extraction from sources the user explicitly subscribes to, we get a bag of words representing the users interests. From these words, the most frequent and descriptive words could be selected as representative of this user. Those keywords can then be compared to a list of trigger words, placing the user in a type. For example, if the words "Ruby", "Java" and "programming" are found as representative of the user, a trigger should put the user in a "programmer"-stereotype. We will have more to say on the subject of keywords extraction later.

In Figure 4.1, the user belongs to three shown stereotypes. Each of these types also has related content from different sources across the user base. In the figure, one source which seems to overlap the user's stereotypes is not explicitly subscribed to. To inform the user of this, a "subscribe" button is inserted alongside the source. Subscribing to the source ensures that the user always gets relevant content from this content provider.

4.1.2 Case Based Recommendations

Case Based Reasoning (CBR) is a process of solving new problems based on the solutions of similar past problems. In AI, these methods are used to solve specific problems or answer queries by computing similarities between a query cases in a database, and sometimes by adapting retrieved cases to the specific query at hand. Mimicking the way humans perform problem solving, techniques from CBR use known cases of problems and solutions to suggest solutions to new problems.

As described by Smyth (2007), methods from CBR are well suited to provide content-based recommendations in domains

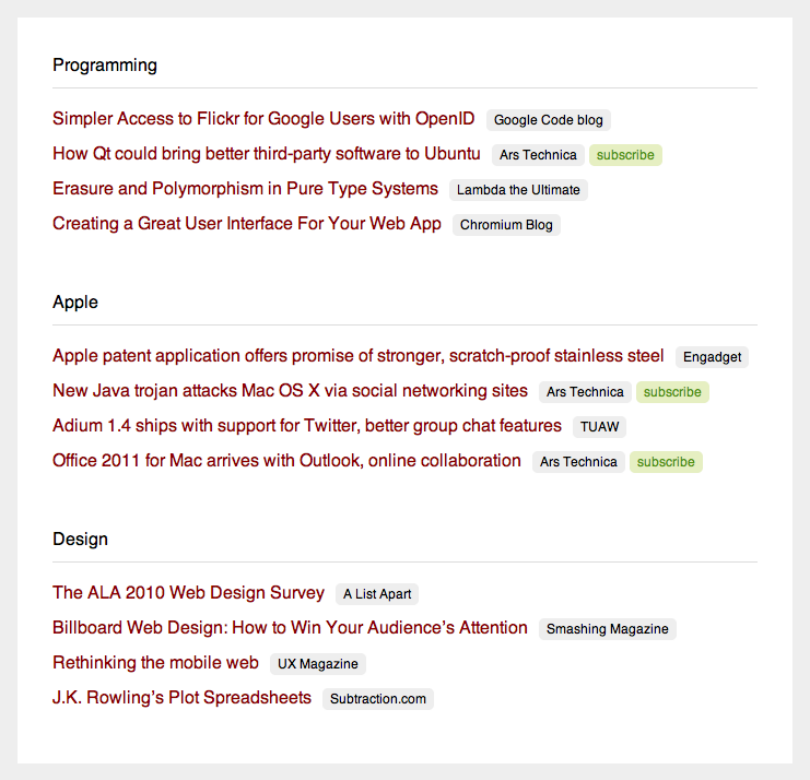


Figure 4.1: **Stereotypes in Signal:** By relating feed items to stereotypes that a user belongs to, items can be presented through helpful categorization. New feeds are presented through supplying new articles and adding a "subscribe" link.

where we have items with well-defined features (e.g. color, price, size).

Smyth (2007) presents a system for recommending digital cameras from a product catalogue. Each "case" is a product, well defined by the features of each camera. The case base is the set of available products. This means we have well defined products in a finite feature space, that allows for structured similarity and retrieval methods.

In Smyth (2007), a query is a structured collection of feature values that is matched against the case database. For instance, the user can specify that he or she wants a camera that costs less than $1000, and that the camera should take images with a resolution of 6 megapixels or more. In systems where we wish to provide recommendations without explicit queries, the recommender algorithm can construct a query based on descriptive implicit or explicit profile keywords. In other words, keywords or values either explicitly provided by the user, or implicitly extracted from observed behavior can form powerful queries.

The important part of a CBR recommender such as this system, is the similarity assessment. This method measures which cases match the query, either directly or implicitly by matching another case similar to the query in question. As the employed similarity measure is what decides how well the CBR method performs, these measures are often domain-specific and even feature-specific.

Smyth (2007) demonstrate a set of feature-specific similarity measures for matching cases with queries and other cases. For example, the similarity of two instances of the "price" feature is calculated by measuring the absolute distance between the two (a symmetric distance measure, see Equation 4.1, where p_t and p_c are prices of two different products).

$$sim_{price}(p_t, p_c) = 1 - \frac{|p_t - p_c|}{max(p_t, p_c)} \tag{4.1}$$

For non-numeric features, the system can use an ontology to infer the similarity of two differing values. In Smyth (2007), the similarity of non-numeric features is estimated from the distance between them in a graph-based ontology.

Smyth (2007) also describe a number of other techniques for improving content-based recommendations with CBR. An important one is the use of feature weights, either globally or individually, to measure the relative importance of the features in a case.

In summary, recommendations based on CBR allows for a flexible number of recommendation methods based on different similarity measures and feature weighting schemes, making them an important consideration when choosing how to filter items based on content.

Dimensions To create personal user models using CBR based recommendations, personal keywords for matching users and documents are needed. This would then be a model-based approach with an individual granularity. Naturally, as the strength of this method is producing structured similarity measures, a system with items that are well structured is where it could provide the most value.

As with stereotypes, the knowledge stored in these models would be long term — personal preferences are in most cases quite stable, although concept drift is still something to remember, meaning that the keywords describing each user can not be a static collection.

If the system allows each user to supply their own descriptive personal keywords, we have explicit knowledge gathering where the blank slate is whatever keywords the user initially supplies. As the method employs structured matching, it would also be able to explain why a certain item gets a certain rating. Indeed, explanation-based CBR is a popular area of research.

The taxonomic placement of this recommendation strategy is given in Table 4.8.

Variable	Value
Predictions	Content-based
Method	Heuristic
Granularity	Individual
Temporality	Long-term
Agents	Explicit

Table 4.8: Taxonomic placement of case-based recommendations.

Case-Based Recommendations in Signal As Signal deals in recommending articles from the web, the level of structure in

these articles would be the deciding factor in whether CBR-based recommendations would give good results. If the system is able to perform high-quality keywords extraction from rated articles (either implicitly or explicitly), we would be able to do accurate estimations of unknown ratings.

Figure 4.2 shows how the presentation of articles can improve if we are able to estimate these ratings. If we know which new feed items will be of the greatest interest to the user, the interface can leverage these ratings in its presentation.

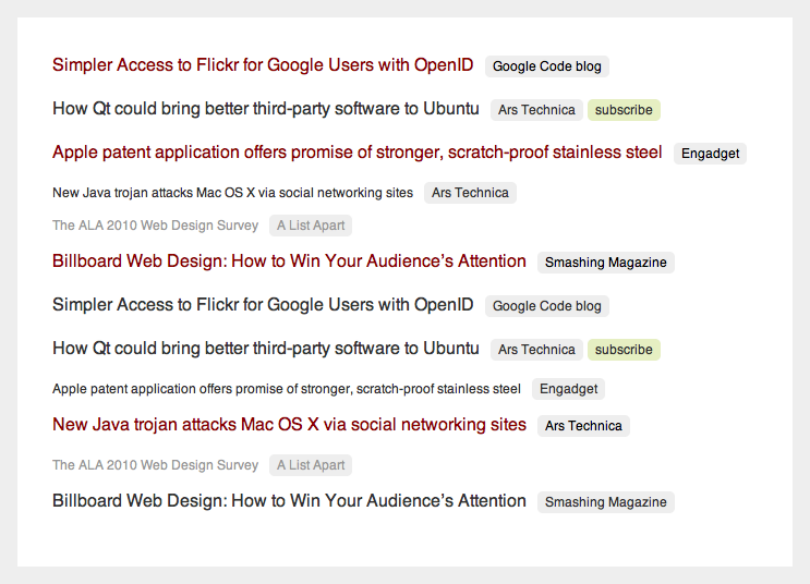


Figure 4.2: **Presentation of rating estimation:** Here, the relevance score of each new article has been calculated. Rather than showing the actual scores, the color and relative sizes of each item is changed to convey their probable importance to the user.

4.1.3 *Semantic Document Modeling*

We always want as much data as possible when estimating ratings of unknown documents. However, as seen, this data is often sparse. When performing content-based modeling, we only consider the documents and each individual user by themselves. Naturally, trying to extract as much information as possible from each document becomes an important task. We will now see how the latent semantic relations in each document has been used to create better recommendations.

Magnini and Strapparava (2001) shows how semantic analysis of documents can be used to create more accurate recommender systems. Their system, SiteIF, analyzes websites visited by a user, and builds predictive user models based on the semantics of the documents. As with the previously described hierarchic stereotypes, this is a content-based, individual approach to providing recommendations.

The original SiteIF system used a simple word-based approach to creating the user models. Each model is represented as a graph, where nodes represent words and arcs

represent co-occurrences between the words. Each node and arch has weights representing their frequency or strength.

Each user has their own graph, which then becomes the words that are most frequent in websites visited by this user. The relevance of a new document can be estimated by computing the relevance of the document to the user model graph.

In their revised SiteIF system, the modeling method is changed to consider the semantic meaning of the visited documents, and not just their actual words. Through this they create a system which provide more accurate estimates of document relevance.

In the new modeling method, documents are represented by *synsets*. A synset is a set of terms, representing an underlying lexical concept. When a new document is visited, the WordNet (WN) synonym repository is used to annotate terms in the documents with their multiple meanings.

The WN repository maps terms to their many possible meanings, and each meaning has a label representing the domain in which the term has this meaning. As WN is a hierarchical graph of synsets, with node relations such as "part-of" and "is-a", the system has a sense of how specific each synset is, and how it relates to other concepts.

When each term is annotated with its synset, a technique for Word Domain Disambiguation (WDD) is used to estimate the actual domain of the document. After the domain has been determined, the highest scoring synsets in the context of the chosen domain is selected to represent the document.

In the new SiteIF system, the user models are represented by the same type of graph, but the nodes now represent synsets. The arcs represent the co-occurrence of synsets. When a new document is properly labeled and annotated with synsets, it is compared and possibly augments the user model. The result of this comparison is an updated model and a relevance score for the document.

In Magnini and Strapparava (2001), the relevance of a document is calculated with their Semantic Network Value Technique:

$$Relevance(doc) = \sum_{i \in syns(doc)} w(i) \times freq_{doc}(i) + \sum_{i,j \in syns(doc)} w(i,j) \times w(j) \times freq_{doc}(j) \qquad (4.2)$$

Here, $w(i)$ is the weight of synset node i in the user model. The arch weight between two synset nodes is $w(i,j)$. $freq_{doc}(i)$ represents the frequency of synset j in the document. Magnini and Strapparava (2001) found a 34% increase in precision and a 15% increase in recall as result of the new semantic user models.

Dimensions In Magnini and Strapparava (2001), the user models represent the interests of users through the semantic nature of their visited documents. SiteIF is primarily described as a personal system running on the user's own computer, meaning that the granularity of the models is strictly limited to the individual in question.

As the characteristics represented by the semantic networks are meant to be long term interests of a user, the temporality of this method is a long-term representation of each user. Concept drift is accounted for through the continuous updating of the models when new documents are considered.

The representation is qualitative in that symbols representing actual concepts are used, and augmented by quantitative measures of the importance of each user aspect.

There are no mention of the starting configuration of user models in SiteIF, but as the semantic graph is dynamically built, it is fair to say that a blank slate would work as a starting point. As with other content-based user modeling methods, this requires some time of usage before proper predictions can be made.

The knowledge acquisition method is a continuous process, where the model is refined each time the user visits a website. The information remains in the model as long as the user visits documents of a similar nature. If synsets in the user model remains at a low score for an extended time, these nodes are pruned from the graph.

As the semantic modeling is a passive knowledge acquisition method, a basic interface utilizing the method needs no explicit means of user input. More important, however, is the output. As this is a recommender system, the ranking and computed relevance of new documents must be displayed in a method understandable to the user.

The taxonomic placement of this recommendation strategy is given in Table 4.9.

Variable	Value
Predictions	Content-based
Method	Model-based
Granularity	Individual
Temporality	Long-term
Agents	Implicit

Table 4.9: Taxonomic placement of semantic document modeling.

Semantic Modeling in Signal Semantic document modeling would be a sure way to increase the precision and recall of our application. Realizing that Signal is at its core in need of automatic recommendation capabilities, using document analysis to rank new articles based on their predicted interest is a compelling possibility.

As this is a purely content-based recommendation approach, the recommendations would be well suited to each user of the application. However, as with all methods using this individual approach, the user would have to use the application for some time to allow the synset graph a chance to get enough data.

Once the model has enough data, the interface can take advantage of the fact that each new article now has a num-

ber representing its overall relevance to the user. Through these scores, the presentation of documents can be visually weighted according to their relevance, as seen in Figure 4.2.

As this method gathers knowledge implicitly through observation of read articles, the interface of Signal does not need any additional controls to facilitate the modeling. As in any application, this can be a welcome feature, if the goal is to minimize distractions perceived by each user.

4.2 Collaborative Methods

This section presents three approaches to collaborative user modeling. Each method uses the social network structure of an application to recommend items, either exclusively, or primarily, augmented by other methods. These methods base their estimations on the intelligence of an often sparse number of explicit user ratings.

We will also look at the different general dimensions of collaborative modeling, and how these methods can be integrated with the interface of our case study application.

Modeling methods in this section	M	G	T	A	#
Billsus and Pazzani (1998): Reducing the user-item dimensionality with SVD (§4.2.2)	M	I	L	E	1,3
Konstas et al. (2009): Social graph traversal with random walks and restarts (§4.2.1)	H	I	L	E	
Ujjin and Bentley (2002): Fine-tuning user profile matching with a genetic algorithm (§4.2.3)	M	I	L	E	

Table 4.10: **Collaborative methods**

4.2.1 *Social Graph Traversal*

Social networks are the explicit and implicit links between users of a system. Explicit links can be synchronous affirmations of bidirectional user relations, or asynchronous directed "fan"-based or "follow"-based relations[2]. Implicit links can for example be the use of collaborative tags or other organizational methods.

Konstas et al. (2009) explores how synchronous user relations in the social network from the music sharing site last.fm[3] can be used to provide collaborative recommendations. They introduce a heuristic, individual user modeling technique based on the Random Walk and Restarts (RWR) graph traversal strategy. The paper also describes how social tags of items can be used to estimate ratings, but as we will have more to say on the topic of folksonomies in Section 4.3.3, this section describes their estimations based on the social network graph.

RWR is a method for measuring the *relatedness* of two nodes in a graph. In Konstas et al. (2009), nodes in the graph are users, tags, and music tracks. Bidirectional weighted

[2] "Friendships" on the Facebook social networking site is an example of synchronous relations, while "followers" on the Twitter social network are asynchronous. See facebook.com & twitter.com

[3] See www.last.fm/about for more information.

edges between the nodes represent relationships and their strength.

Starting from a node x, the RWR algorithm randomly follows a link to a neighboring node. In every step, there is a probability α that the algorithm will restart its random walk from the same node, x. A user-specific column vector $\mathbf{p}^{(t)}$ stores the long term probability rates of each node, where $\mathbf{p}_i^{(t)}$ represents the probability that the random walk at step t is at node i. $\mathbf{S}$ is the column-normalized adjacency-matrix of the graph, i.e. the transition probability table. $\mathbf{q}$ is a column vector of zeroes with a value of 1 at the starting node (that is, $\mathbf{q}_i$ is 1 when the RWR algorithm starts at node x). The stationary probabilities of each node, signifying their long term visiting rate, is then given by Equation 4.3.

$$\mathbf{p}^{(t+1)} = (1-\alpha)\mathbf{S}\mathbf{p}^{(t)} + \alpha\mathbf{q} \tag{4.3}$$

This algorithm is repeated until the values of $\mathbf{p}$ converge to within a small delta. Then, the *relatedness* of nodes x and y is given by $\mathbf{p}_y$ where p is the user model for the user represented by node x.

Konstas et al. (2009) found that their approach outperformed a simple baseline collaborative recommender based on the *Pearson correlation coefficient* (a statistical measure of the linear dependence of two variables. See Equation 4.4). However, they also found that in order to create accurate rating estimations from social networks, the user relationships must be a significant part of the system. That is, the algorithm performs better when many precise relations between users are represented. As last.fm is a site where the social network only plays a supportive role, they suggest that their algorithm would perform better on a site with more social connections.

Dimensions In this heuristic approach, the social graph is key to estimating unknown ratings. There are no explicit user models, just the algorithm that traverses the graph. If anything, the user-specific vectors can be said to be a sort of user model.

Variable	Value
Predictions	Collaborative
Method	Heuristic
Granularity	Individual
Temporality	Long-term
Agents	Explicit

Table 4.11: Taxonomic placement of social graph traversal.

The starting configuration of this method relies on a blank canvas, requiring users to form explicit social bonds before any recommendations can be made. On the other hand, a system based on traversal of the social graph would not need a lot of connections for a new user, since even one connection would allow the algorithm access to a lot of other nodes. However, as found by Konstas et al., the more connections the better.

The knowledge gathering agent is indirect in that users create explicit social connections for other reasons. Social

relations can be created simply to view other people's profiles to see how they use the system in question. Of course, the knowledge gathering can also be seen as explicit, if users are presented with a system where the sole purpose of forming connections is to provide recommendations. Either way, the interface would have to provide mechanisms for creating and managing social connections.

The taxonomic placement of this recommendation strategy is given in Table 4.11.

Social Graphs in Signal Relating the findings Konstas et al. to Signal, it seems clear that for collaborative modeling to work, the social aspect would have to be an important and prominent part of the user experience. Accordingly a method such as this demands a lot from the application interface.

The social aspect would also change the nature of our application in a fundamental way. As demonstrated with the last.fm example, sites that add a social context as an auxiliary and purely additional feature of their service would not get as good results as one where the social aspect is paramount, e.g. in the aforementioned services such as Facebook or Twitter.

A welcome aspect of social modeling is that explanations of recommendations becomes both easy to do and helpful. By labeling recommended items by which socially connected other user this is recommended because of, the social fabric is utilized to give users good reasons to read a new article. For example, if an article is recommended to a user because a good friend rated it highly, this social explanation might be especially valuable.

4.2.2 *Recommendations as a Classification Problem*

Billsus and Pazzani (1998) argue that collaborative recommender system do not leverage methods from machine learning as well as they should. According to them, while many systems perform the same tasks that many ML methods do, they rely too heavily on the simple problem-specific evaluations of the ratings matrix.

According to Billsus and Pazzani (1998), simple reliance on the ratings matrix in collaborative recommender systems raises a couple of problems: First, correlation between two user profiles can only be computed based on rated items the two profiles have in common — often a very small number. Second, these methods are often restricted by their simple, global similarity metric that only measures similarity through corresponding positive ratings. For example, these methods will not be able to identify a situation where negative ratings from user A perfectly predict positive ratings for user B.

	I_1	I_2	I_3	I_4	I_5
U_1	4		3		
U_2		1		2	
U_3	3	4	2		4
U_4	4	2	1		?

Table 4.12: Matrix of items, users and ratings. We wish to estimate the rating user 4 gives to item 5.

To solve these problems, collaborative filtering can be seen as classification problems — a topic well studied in the field of ML. Individual user models are created by supervised construction of classifiers for each user. The training data for these classifiers are the items that have ratings from the user in question. The features for these items are the ratings from other users.

For example, consider the ratings matrix in Table 4.12. We wish to estimate how well user 4 will like item 5, i.e. $\hat{u}(u_4, i_5)$. To create a classifier for user 4, we only consider items explicitly rated by this user. Then, we create discrete feature values for these items by transmuting the ratings from other users into either 1, if the user liked the item, or 0 otherwise. The correct class that the classifier should identify is computed from the rating given by the user we wish to create a classifier for (e.g. user 4). See Table 4.13.

By transmuting the ratings matrix into feature vectors like those given in 4.13, we may now use any supervised classification method from the field of machine learning.

Billsus and Pazzani (1998) suggest that to cope with the huge number of dimensions in each feature vector, an algorithm that can reduce this number should be used. This is very similar to problems faced in document classification, where documents with a large number of words must be processed in some way. Latent semantic indexing (LSI) is a popular ML technique for text classification that utilize singular value decomposition (SVD) to reduce the number of dimensions and rather rely on latent relations found in the text. While a through discussion of LSI and SVD is beyond the scope of this section, suffice to say that these method greatly reduce the number of items in the feature vectors, allowing for efficient construction of classifiers.

In their example, Billsus and Pazzani use a neural network trained through back-propagation with the reduced feature vectors to create the final classifiers. To predict the utility of a new, unseen item, the items ratings are converted to a boolean feature vector. This vector is then scaled to the dimension of the reduced feature matrix, and fed into the neural net which produces the final rating.

	E_1	E_2	E_3
U_1 like	1	0	1
U_1 dislike	0	0	0
U_2 like	0	0	0
U_2 dislike	0	1	0
U_3 like	1	1	0
U_3 dislike	0	0	1
Class	like	dislike	dislike

Table 4.13: Training examples for the classifier of U_4, with discrete feature vectors.

Variable	Value
Predictions	Collaborative
Method	Model-based
Granularity	Individual
Temporality	Long-term
Agents	Explicit

Table 4.14: Taxonomic placement of classification-based recommendations.

Dimensions The framework presented by Billsus and Pazzani is a powerful idea. If recommender systems could easily tap into the knowledge and research invested in text classification from ML, a lot of interesting methods can be used. The user models in this setup are the personal classifiers constructed for each user, no matter what method is used to construct them.

With an individual granularity, and a good strategy for coping with new, unknown users through collaborative fil-

tering, using classifiers would seem like a good fit for many services. The collected knowledge is of long-term value, although the classifiers would have to be retrained on a regular basis to deal with concept-drift. This might make the method prohibitively expensive in settings with a lot of users, unlike other, often hybrid, recommender methods.

Other than some mechanism for rating items, the interface for such an approach would need little in the way of methods for user interaction. The learning methods estimate new probabilities, not through explicit social relations, but via implicit social similarities based on matching ratings. This ensures that the recommender system would not require a lot of work on the part of each user.

The taxonomic placement of this recommendation strategy is given in Table 4.14.

Collaborative Filters in Signal As with all out methods that primarily estimate ratings for unseen items, the resulting interface for Signal gain two main opportunities. First, the use of relevance when displaying items from the user's subscribed sources. As previously seen in Figure 4.2, when we estimate ratings for unseen items, these can be displayed in a way that shows off their differing estimated rating for the active user.

Second, if our goal is to recommend articles from sources that the user has not subscribed to, i.e. recommend articles from unknown websites, we can use the approach visualized in Figure 4.6. Here, new items enter the stream based on collaborative filtering, with new and possibly relevant items clearly marked as such.

Of course, using both approaches would result in a quite interesting application. Not only would every received article be displayed according to its estimated relevance, but new items would also arrive, possibly expanding the user's interests, while at the same time showing just how interesting these new articles may be.

4.2.3 *User Matching With A Genetic Algorithm*

Can biologically inspired methods from the field of sub-symbolic AI be used to provide recommendations? Let us examine one approach[4].

In "Learning User Preferences Using Evolution", Ujjin and Bentley (2002) describes a model-based collaborative recommender system based on a genetic algorithm[5] (GA). Their heuristic and collaborative approach, used to recommend movies from an online movie site to users, a GA is used to improve the quality of user profile matching.

Before users can be compared, the system builds a collec-

[4] Floreano and Mattiussi (2008) gives an introduction to methods in artificial intelligence directly inspired from natural phenomena, such as genetic algorithms, swarm intelligence and cellular and neural systems

[5] A genetic algorithm is a sub-symbolic and biologically inspired AI method. These algorithms use principles from natural evolution to find solutions to problems. A GA creates a number of *individuals* (possible solutions) that may or may not pass their traits onto the next *generation* of individuals. At each generational shift, the individuals are combined and mutated, based on their estimated fitness (how good the solution is), just like in nature. After a number of generations, the best individual should represent a good solution to the problem.

tion of profiles for each user, representing their movie preferences. Each profile represents a user's preference for a movie. A profile consists of 22 features, containing the rating, and user-specific features like age, gender and genre frequencies (see Table 4.15).

When the individual user profiles are created, the next task is profile selection, i.e. selecting the set of profiles that the active user will be compared to. Like many systems that have to scale to accommodate a large number of users, Ujjin and Bentley (2002) use a random sampling of the user base.

After selection comes profile similarity computations, which begins by using the genetic algorithm. The 22 different user- and movie-specific features from each profile vectors are weighted to properly define each user's priorities. An individual in the GA is one profile, $p(u_j, c_i)$.

The GA creates a set of random individuals, or genotypes, which are combined and randomly mutated through evolutionary generations. Each new genotype is developed into a phenotype, which in our case is the vector of profile feature weights. In Ujjin and Bentley (2002), the top performing 40% of the individuals are kept across generations. To identify the best solutions we have in each generation, the fitness of each individual phenotype (i.e. how good the weights are), is calculated.

Rating	Age	...	Genres
5	23	...	001100100..

Table 4.15: **User profile:** An example of an individual profile for user i of movie j. The (...) represents 19 other features that each profile retains to describe the attributes of a user and an item.

As in any GA, this fitness function is the most complicated and important step to achieve good solutions. The fitness function used in this movie recommendation system is defined as a supervised learning task. All the movies rated by each user are randomly partitioned into two sets for training and testing data. To calculate the fitness of an individual set of weights, the system first finds a set of neighboring users based on a simple euclidean distance measure of normalized versions of user profiles, weighted by the evolved phenotype. The ratings for the movie in questions by closely matching users are then indicative of the fitness of our evolved set of weights.

When the weights for each profile have been fine tuned through evolution, these weights are used together with the selected user profiles to compute the similarity of users. This is done by using a weighted euclidean distance computation, that measures the distance between two user based on their weighted features.

We now have a set of the best matching neighborhood of users to our active user. Recommendations are made by selecting the highest rated unseen movies in this set.

Ujjin and Bentley (2002) found that using the weights evolved by the GA made more accurate similarity measures than the alternative of using the same euclidean distance metric without weights. In other words, by fine tuning the weight

of each profile feature, the system made better recommendations.

The researchers continued their work in Ujjin and Bentley (2003), by substituting the GA for another sub-symbolic AI method: Particle Swarm Optimization (PSO)[6]. The PSO version was found to achieve marginally better results than the GA approach, but more importantly, was able to compute the similarities significantly faster, resulting in a recommender system more suited for large collections of users and items.

[6] Particle Swarm Optimization are methods that use iterative computations based on the nature of particles. Each particle represents a possible solution. Particles move about the space of possible solutions (the search space), and each position has a specific utility for each particle, resulting in particles (possible solutions) that move towards hopefully global maxima in the search space.

Dimensions The use of sub-symbolic and biologically inspired AI methods in recommendations is an interesting approach, not only because of the inherent excitement of these methods themselves. While symbolic methods may have gotten more attention from researchers, using sub-symbolic methods may give quite positive results, as seen in this section.

In this approach based on a GA, the user models are the many phenotypes that represent each user's preference for a certain set of movies. Granularly, this is then an individual approach. The knowledge is long term, and might not be as hurt by concept-drift, as the ratings given by the user models are quite separate from each other. That is, while the interests and ratings of one user might change over time, the collective hive mind interest of the entire user base should be more stable.

As with all collaborative recommender methods, this approach is susceptible to the problems of rating entirely new items, the problems of sparse rating data, and recommending items to users that do not align with the mainstream opinion of the main user base. See Section 4.3.1 for a discussion on these weaknesses.

The taxonomic placement of this recommendation strategy is given in Table 4.16.

Variable	Value
Predictions	Collaborative
Method	Model-based
Granularity	Individual
Temporality	Long-term
Agents	Explicit

Table 4.16: Taxonomic placement of evolutionary recommendations.

Evolving User Models in Signal Despite the somewhat esoteric approach taken by Ujjin and Bentley, this evolutionary recommender can be used in Signal on the same terms as other recommender systems. As long as the produced estimates are viable, the question come down to how the method deals with inherent problems of user modeling (e.g. concept drift), and its computational complexity.

As mentioned, the PSO-based method outperformed the GA in a decisive manner. In an application such as Signal, with a potentially large database of both users and articles, the expected runtime of each method would be a critical factor in determining its viability. Accordingly, the GA-based method might not be the best for real-time and continuous estimation

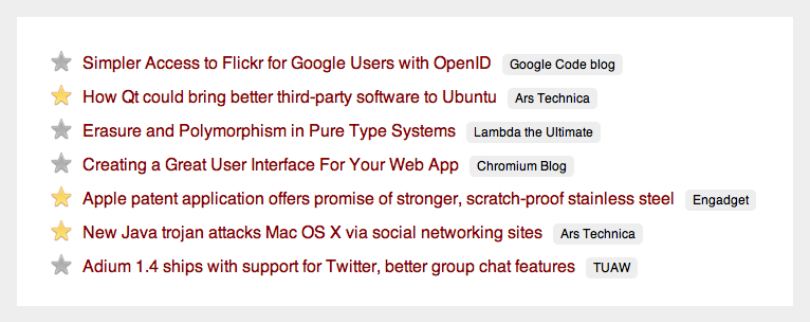


Figure 4.3: **Assigning ratings in Signal:** Explicit ratings can be assigned by a simple binary choice for users. In this example, if a user clicks the star beside an article, this indicates a positive rating. This can also be combined with extra functionality, by allowing users to retrieve only their starred items, when they are looking for something previously found interesting.

of new ratings. Figure 4.3 shows one way to collate ratings in an application such as Signal.

As with other recommenders that primarily estimate the interestingness of unseen items, the interface of Signal could leverage this in a number of ways. As discussed, this is shown in Figures 4.2 & Figure 4.6.

4.3 Hybrid Methods

This section looks at hybrid recommendation methods that combine both content-based and collaborative approaches to gain the strength of both and problems of neither. These methods are most applicable in systems with both data-rich items and a social networking aspect.

We will also look at the different general dimensions of hybrid modeling, and how these methods can be integrated with the interface of our case study application.

Modeling methods in this section	**M**	**G**	**T**	**A**	**#**
Claypool et al. (1999): Weighted average of content-based and collaborative methods (§4.3.1)	H	I	L	E	1,3,4
Hotho et al. (2006): Vector space model with folksonomy-based similarity measures (§4.3.3)	H	I	L	E	
Huang et al. (2002): Graph traversal through a shared user-item graph (§4.3.2)	H	I	L	I	

Table 4.17: **Hybrid methods**

4.3.1 *Weighted Recommender Combination*

Purely content-based or collaboration-based approaches to predicting user ratings have a broad spectrum of problems.

In their effort to implement personalized recommendations for an online newspaper, Claypool et al. (1999) describes a number of problems with choosing a pure approach. First, while collaborative methods add a level of human intelligence to the estimations, the lack of ratings for new items added to the system limits their precision for recommending news. This is called the *early rater problem*. Collaborative methods must also face the problem of sparse rating data, as the number of items and users far exceeds the number of ratings. In addition, some users will have opinions that oppose those of

the main user base, rendering mainstream recommendations useless. This is called the *gray sheep problem*.

Content-based approaches, while often effective and able to use the entire collection of items, have their own problems. Most importantly, computers have a hard time estimating the quality of information, at least compared to human evaluations.

To mitigate these problems, Claypool et al. (1999) take a hybrid approach, where two separate recommenders are combined in a weighted average[7]. In other words, their system for recommending news articles first perform content-based rating estimation, then estimate a collaborative rating, before finally combining these values. The values are each weighted on a per-user and per-item basis, adding further strength to their approach.

[7] Claypool et al. (1999) claim their approach is not a hybrid recommender, as the two modules are completely separated. However, as the final rating estimations are based on both the content of items, and ratings from similar users, this technically sound definition is debatable.

The collaborative part of their approach is based on the *Pearson correlation coefficient* (seen in Equation 4.4). The prediction is computed as a weighted average of the ratings given by users similar to the active user.

$$prediction(u_i, c_i) = \bar{u}_i + \frac{\sum_{j=1}^{n} corr(u_i, u_j) \times (rating(u_j, c_i) - \bar{u}_j)}{\sum_{h=1}^{n} corr(u_i, u_h)} \tag{4.4}$$

In Equation 4.4, u_i is the active user and c_i is the item to estimate a rating for. $\bar{u}$ is the average rating for the user in question, $rating(u, c)$ is the rating a user has given to an item, and $corr(u, j)$ is Pearson's correlation coefficient, a statistical measure of the linear dependence of two variables.

The content-based approach matches users with items based on a number of keywords stored in each profile. These keywords are taken from explicit topic selection by users, and descriptive words mined from articles each user gives a high rating. The items are then matched with users by computing the *overlap coefficient*[8] of article and profile keywords.

[8] The overlap coefficient M is computed by the following formula:

$$M = \frac{2|D \cap Q|}{min(|D|, |Q|)} \tag{4.5}$$

D is the set of keywords from an article, Q the set of profile keywords of the active user.

The two recommenders are combined by weights estimated on a per-item and per-user basis. These weights start out with an equal share, and are adjusted when users explicitly rates items. The absolute error between the predicted and given rating is computed, and used to adjust the weights. The weights fluctuate quickly in the beginning, and stabilize according to user interests as more and more explicit ratings are given.

Claypool et al. (1999) found that their approach outperformed both purely content-based and collaborative methods, especially over time as each user provided more ratings.

Dimensions Classifying a hybrid method such as the one described by Claypool et al. is not as straight forward as some

of the other methods described so far. The methods described here are heuristic approaches that estimate ratings based on simple formulas. On the other hand, by constructing sets of keywords as user profiles, there is a model-based aspect to this as well.

Clearly, the knowledge gathering process is explicit. The system uses explicit ratings provided by users to estimate new ratings, and the personalized keywords can be explicitly provided as well. As with any explicit knowledge gathering method, this requires an extra effort from users if they wish have personalized recommendations.

As this framework does not specify a certain method, but rather a way of using any classification-based method from ML, the taxonomic placement will be dependent on the method of choice.

The taxonomic placement of this system when using the described classifiers is given in Table 4.18.

Variable	Value
Predictions	Hybrid
Method	Model-based
Granularity	Individual
Temporality	Long-term
Agents	Explicit

Table 4.18: Taxonomic placement of hybrid weighted recommenders.

Hybrid Weights in Signal The resulting interface from using hybrid weights would be a lot like the one seen in Figure 4.2. Each article can be displayed in a way that corresponds to its estimated rating.

If we were to add the possibility of explicit rating, we would need some extra mechanism in the interface. The simplest method would be to add a "star" or "thumbs up" icon next to each article. While this would only produce binary ratings, the thought of having each user enter scalar values to rate articles would surely put a few users off the system. An example of these types of ratings can be seen in Figure 4.3.

Of course, using two distinct recommenders in this form would also necessitate a social layer in our application. This would require a more complex interface where users can form social relations, by seeing other profiles and adding them to their network. We will discuss this increase in complexity in Chapter 5.

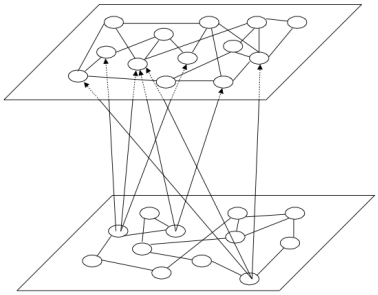


Figure 4.4: **Recommender graph:** The recommender system builds a graph of document and user nodes, with links between nodes of the same type describing similarity, and links between nodes of different types representing ratings or purchases. Figure from Huang et al. (2002).

4.3.2 *Graph-Based Hybrid Recommender*

Huang et al. (2002) presents a system where both each user's personal actions, and those of similar users, are taken into account. The purpose of their system is to recommend books from a digital library.

In their approach, the recommender system is an explicit two-dimensional graph structure (see Figure 4.4). In the first dimension, the nodes are the items available for recommendation, e.g. products, books or documents. Edges are drawn between the nodes by computing item similarities through traditional methods, for example by computing document similarity with TF-IDF[9].

[9] See Baeza-Yates and Ribeiro-Neto (1999) for an introduction to methods from the field of information retrieval and document modeling, including classic methods such as TF-IDF.

In the second layers, the nodes represent users. Each node is an individual user, and links are drawn between user nodes based on their similarity. In Huang et al. (2002), user similarity is computed by computing the overlap of the values of fields that represent each user.

As seen in Figure 4.4, there are also links between user and item nodes. These links represent explicit ratings between users and products. This can mean that a user likes a document, has bought a product, et cetera.

Every link in the 2D graph is weighted. In the document and user graphs, weights are used to indicate how similar two items or users are. The weights on links between users and items indicate the strength of the recommendation from the user to the item. A condensed example of a small user-item graph can be seen in Figure 4.5.

Recommendations are provided by traversing the graph while accumulating the weights. The approach can be customized by setting how many levels of association the system should allow.

To achieve a content-based recommender, the graph is traversed from a user node to an item node via a recommendation link from this user, and then to similar items as indicated by links in the item layer. For example, referring to Figure 4.5, the 2-degree association of the path $C1 \rightarrow B2 \rightarrow B1$ would give a content-based recommendation of book B1 to user C1. This score can for instance be calculated by multiplying the weights along the path:

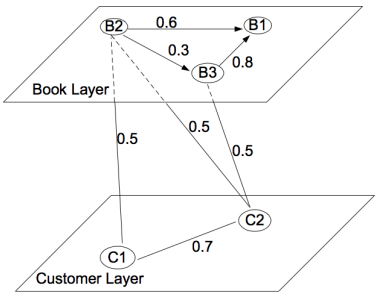


Figure 4.5: **Graph traversal recommendation:** To recommend items to users, the graph is traversed starting from the user in question, going through user nodes for collaborative retrieval and through book nodes for content-based retrieval. Figure from Huang et al. (2002).

$$AssociationScore(C1, B1) = w(C1, B2) \times w(B2, B1) = 0.5 \times 0.6 = 0.3 \tag{4.6}$$

If we allow higher degree associations, the recommender algorithm will find more paths between the user and the document. If these paths go through other user nodes, we will in effect have a hybrid recommender, using both content-based and collaboration-based paths. For example, with 3-degree association there will be three more paths between $C1$ and $B1$, as seen in Table 4.19. By traversing all the possible paths and summing the resulting *AssociationScore* values, we get the final recommendation score for item $B1$ for user $C1$.

In Huang et al. (2002), low-degree recommendations were achieved by simply fetching items similar to those already rated by the user. High-degree association recommendation is treated as a concept retrieval task, and a Hopfield Net algorithm is used to perform a parallel relaxation search. The graph is traversed until the activation levels, or weights, achieve convergence. When the network has converged, the algorithm returns the top items not already seen or purchased by the user as a result.

Association	Possible paths
2-degree	$C1 \rightarrow B2 \rightarrow B1$
3-degree	$C1 \rightarrow C2 \rightarrow B2 \rightarrow B1$
3-degree	$C1 \rightarrow B2 \rightarrow B3 \rightarrow B1$
3-degree	$C1 \rightarrow C2 \rightarrow B3 \rightarrow B1$
$AS = 0.3 + 0.21 + 0.12 + 0.28 = 0.9$	

Table 4.19: **Association paths:** By increasing the association level, more paths between a user and an item can be found. However, if the level is set too high, the computational complexity becomes too great. AS is the final association score between $C1$ and $B1$. These paths correspond to the network in Figure 4.5.

Huang et al. (2002) found that the hybrid approach outperformed both the purely content-based and collaborative approaches with respect to precision and recall metrics. However, performing higher level association retrieval through use of the Hopfield algorithm provided no significant improvement.

Dimensions The user models in this graph-based approach are the user nodes and their respective links[10]. While the granularity of these models are individual, all users reside in the same central structure, both to allow links to the same item graph, and to allow for collaboration-based recommendations. The temporality of this model is long-term as links between users and items remain over time. However, the model is well suited to mitigate concept drift, as the weights are scalars that can be tweaked when new knowledge is gathered.

[10] Using graphs to describe recommender systems is a popular approach, even if the underlying system does not employ any explicit graph structure. Mirza et al. (2003) describes how any recommender system can be described by its implicit graph that allows jumps between users and items.

The knowledge acquisition process begins with the description of each user node. If user information fields are present, each users provide some details about themselves before recommendations can be made. If not, similarity links between user nodes can be inferred by calculating how many links they have in common to the same item node. Needless to say, this would require some interaction between the user and one or more items before recommendations can be made. On a positive note, once some information about the user is known, the gathering agent is completely implicit, and the user does not have to explicitly provide information to the recommending system.

Variable	Value
Predictions	Hybrid
Method	Heuristic
Granularity	Individual
Temporality	Long-term
Agents	Implicit

Table 4.20: Taxonomic placement of hybrid graph-based recommenders.

As for interface integration, this system would be well suited for seamless integration of a "recommended items" approach, or simply for ranking items in a collection. The method is also well suited to present explanations for recommendations, on the form "item X is recommended because you purchased items Y and Z" or "similar users recommend item X".

The taxonomic placement of this recommendation strategy is given in Table 4.20.

Graph Traversal in Signal The graph-based hybrid recommender would be a suitable addition to our case study application. As the method allows for simple ranking of documents based on reading history, articles could be presented in an order that matches what the user wants to read. Accordingly the user-user links could easily be computed by connecting users that over time read the same articles.

If we wish to add a more visible social level to our application, both new articles and other users can be recommended. Imagine a setting where the recommendation system dis-

covers two users with very similar reading patterns. Recommending either a symmetrical or asymmetrical association to each user would seem like a valuable addition.

As the possibilities of ranking documents through this user modeling method mimics those described in *semantic document modeling*, the resulting interface could be presented in the same manner, as seen in Figure 4.2.

4.3.3 *Folksonomy-Augmented Models*

Web applications often have more information about users and items (documents, sites or articles) than simple ratings. One of these extra resources are tags, simple keywords assigned from users to items. The collection of users, items, tags and user-based assignment of tags to resources is called a *folksonomy*[11].

[11] The term *folksonomy* is a rather clumsy combination of the words *taxonomy* (categorization) and *folk*. Whenever users can tag items with their own terms, we have a folksonomy.

Hotho et al. (2006) defines a folksonomy as a tuple $\mathbb{F} = (U, T, R, Y, \prec)$. Here, U, T and R are finite sets of users, tags and resources (items), respectively. Y is a ternary relation between users, tags and resources, called tag assignments. $\prec$ is a user-specific tag hierarchy, applicable if the tags are organized as super- and sub-tags. The *personomy* $\mathbb{P}_u$ is a user-specific part of $\mathbb{F}$, i.e. the tags, items and assignments related to one user u. In our terms, this personomy would be the user model. Hotho et al. use folksonomies to do information retrieval based on their FolkRank search algorithm, a derivative of the famous PageRank algorithm (Page et al., 1998). However, folksonomies can also be used to augment standard user-item based recommendations.

Bogers and van den Bosch (2009) examine how folksonomies can be used to improve recommendations on social bookmarking websites, i.e. websites where users can bookmark resources and annotate these links with tags[12].

[12] Social bookmarking websites are prime examples of content aggregating websites. Two examples are URL bookmarking through delicious.com or bookmarking of academic papers with services such as CiteULike.org

In Bogers and van den Bosch (2009), recommendations based on folksonomies are compared with regular user-user and item-item similarity computations. The graph of users, items and tags form a 3D matrix, where each cell corresponds to a tag assignment from a user to an item. From this 3D matrix, two 2D matrixes are extracted: The user-tag matrix and the item-tag matrix.

Their first approach is doing content-based recommendations by computing item similarities through tag overlap. Each item that is in the collection of the active user u_k is compared to other items that the user has not yet added. The similarity between two items can be computed using common similarity metrics, for example the Jaccard overlap or Dice's coefficient. The final ordered set of recommended items are the items that has the highest similarity score with the items the user in question has already added.

Their second approach is to use folksonomy-based similarities of users to recommend new items, creating a collaborative recommendation system. The similarity between two users is calculated by their common tag overlap, just as in the item-based approach described above. The set of items to rank is the set of unseen items added by the users most similar to the active users. These items are then ranked from the sum of user similarity measures.

Bogers and van den Bosch (2009) compared these two new similarity metrics to more common metrics that did not consider the folksonomy. These common metrics computed user-based and item-based similarities by using the cosine similarities between user-item and item-user vectors, respectively.

The results are quite interesting: For computing the similarity of items, the tag overlap similarity considerably outperformed the traditional approach. This is not surprising, as their work show that, at least in their datasets, item-tag vectors are significantly less sparse than item-user vectors. On average, the number of tags assigned to an item was 2.5 times higher than the number of users who have added the item.

Interestingly, while using tags to improve item similarities greatly improved the results, using tags to compute the similarity of users gave worse results than the traditional approach. This can be explained by the same logic: When users have more items than tags on average, using the user-item vectors will provide more data to use in a similarity measure.

In other words, tag-based similarities can greatly improve the recommendations in a content-based approach, but does not seem to help when using a collaborative approach.

Dimensions The user model in a folksonomy-based approach is the *personomy* $\mathbb{P}_u$, of each user. The personomy represents the tags, resources and tag assignments that each user has made, and is then a collection of items of interest to this user, where the actual interest is described through keywords. The granularity of this approach is obviously individual, as each user has their own personomy. Temporally, the models store the whole history of interaction, so to compensate for concept drift, tags may be have to be prioritized by the timestamp of their last usage.

Variable	Value
Predictions	Hybrid
Method	Heuristic
Granularity	Individual
Temporality	Long-term
Agents	Implicit

Table 4.21: Taxonomic placement of folksonomy-augmented recommendations.

The knowledge gathering process is all about tags. The models start with a blank slate and become more descriptive as each user adds more tags. This is, in essence, an explicit form of gathering user modeling data, although tags are often used on sites for other purposes, e.g. providing a way for users to personally index their saved items.

The interface integration is quite simple. The application has to allow for tagging of items, and removal of tags. In addition, to prevent the space of tags from becoming too

large, suggesting more appropriate tags through stemming or lemmatization would be a welcome part of the interface.

The taxonomic placement of this recommendation strategy is given in Table 4.21.

Folksonomies in Signal There are basically two approaches where recommendations from tags could be used in signal: First, as the method described is to recommend new items that the user has not already seen, one approach would be to recommend new items to each user that do not come from any explicitly added source.

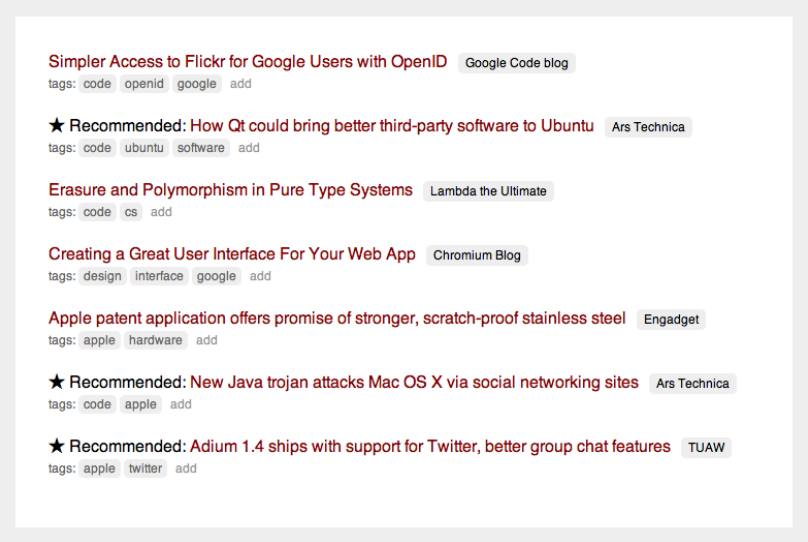


Figure 4.6: **Folksonomy in Signal:** The two described possibilities of using the methodology explained in this section is showed in this concept mockup. Already present items can be ranked according to tags, and new items can be recommended. In this example, the unseen items are marked as "recommended", to impart that these articles do not come from one of the user's explicitly added sources.

Since this method can also be used to rank documents in a collection, the approach can also be used to rank articles that come from the user's sources. While this is a different approach than what is described in the examined paper, there would have to be some changes.

First, we can not defer the ranking of an item *until* the user has tagged it — the item would already have been considered by the user, making the new ranking worthless. A better way is to see what other users of the application has tagged the new article with, and use these tags in accordance with the active user's tags to rate the item. This can also help in suggesting tags the active user should apply to the ranked item.

An interface using both approaches, to serve new unknown items as well as ranking existing items is demonstrated in Figure 4.6.

Concluding Thoughts

The previous chapters have given an overview of what user modeling on the Web entails. It is now time to draw parallels and find common ground shared by the widely differing methods. This chapter also evaluate what has been presented on user modeling in our case study, and what future work might be beneficial.

5.1 User Modeling on the Web

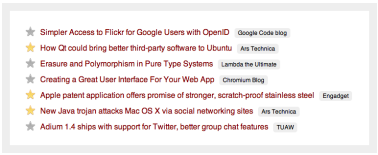
What have we learned from our exploration of user modeling methods? First of all, that user modeling is a complex subject with as many aspects as there are different methods trying to solve the problem.

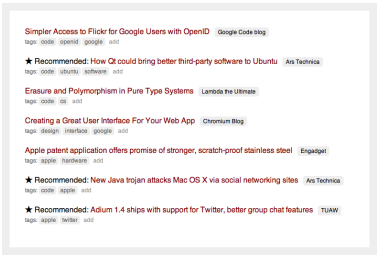
While our discussion of modeling methods described the topic from a bottom-up perspective, there is little doubt that when considering different approaches to adaptive web content, a top-down approach should be taken. Starting with the user interface, a developer would have to identify the exact value provided by implementing user modeling. Only after this end-user value has been established does it make sense to begin comparing different modeling methods to each other.

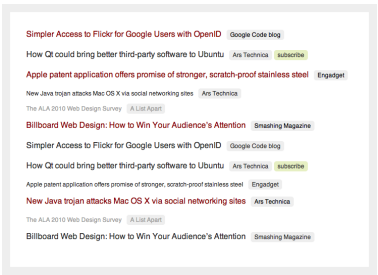
When the possible value of such a system has been explored, there has to be clarity in what data should be used to perform estimation of rating. Does the application in question have well structured information about items? Are there, or should there be, a social aspect to the service? Can we use both the content and social connections to estimate unknown ratings?

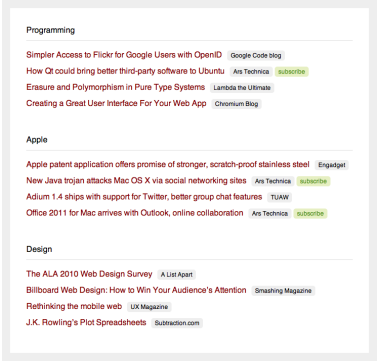
As demonstrated in our exploration, both content-based and collaborative recommender system have their own unique weaknesses. The hybrid approach would then seem to be the most promising. However, as demonstrated by the different applications each method is a part of, there is no one correct answer to the question of what data to leverage — this answer can only be given in the context of a specific application.

Clearly, the more data we have on each user, the more precise estimations of ratings can be performed. By using a combination of content-based and collaborative techniques, each method's weaknesses can be mitigated. However, as seen in our discussion on interface integration, collecting more data is not always beneficial to the users. For example, while a social aspect might result in better recommendations, the added interface complexity of social connections might result in a









system of less value to the end users. Additionally, if the use of auxiliary content like folksonomies and social relations are orthogonal to the actual service provided by an application, users might be reluctant to invest time in features not directly relevant to their use case.

When considering the algorithmic similarities of the methods explored in the previous chapter, graph theory emerges as a popular way of describing and implementing recommender systems. This is a natural choice: A graph is a direct representation of how the resources of such a system interacts. The systems have items and users. These are related to each other with various strengths. The act of providing recommendations is then well described by exploring how this graph is traversed (see Mirza et al. (2003)).

The other dominant technique extracted from our review is the user-rating-item matrix, where users and items are mapped by ratings. Most hybrid techniques rely on formulas that directly access and compute ratings from this matrix. Naturally, maintaining an actual matrix of users and items would be unfeasible in a system with a large number of users and items, as well as wasteful considering the sparsity of ratings. Accordingly, these methods would have to rely on methods from the field of information retrieval, like dictionaries and posting lists.

While the explored methods display a range of techniques for estimating ratings, the framework for mapping recommendations to methods from machine learning (Billsus and Pazzani, 1998) is an especially interesting approach. By bridging the two fields, many interesting combinations can be used for estimating ratings.

In any event, user modeling on the Web is highly application and service specific, where the needs of users in one particular service must be the guiding light before diving into the gritty details of choosing predictive algorithms and knowledge gathering methods.

5.2 User Modeling in the Case Study

What have we learned about user modeling in the context of our own content aggregating web application? The notion of information overload is an especially fundamental problem in a service such as Signal. The presentation of a wide variety, and possibly staggering amount of content can certainly improve by tailoring the application to each user. There is no doubt that feed readers, with their goal of syndicating a broad range of differing websites are prime candidates for inflicting information overload.

As we have seen, the different modeling methods can give

us many exciting possibilities in improving the interface — from displaying each article according to its estimated rating, to inserting new unseen articles in the information stream, hopefully providing extra value to the users.

On the other hand, it should also be clear from our discussion of the usability of autonomy that each initiative taken by the application raise a number of questions. From the issue of transparency, to the issue of privacy — there is no clear-cut method of user modeling that will improve the end user experience.

In addition, especially when considering collaborative approaches to recommender systems, powerful inference requires a lot of data — data that often has to be explicitly provided by each user. As our goal is enhanced usability, the tradeoff between more interesting content and having to supply extra information is a central consideration when adding personalization to an application such as Signal.

There is also the question of granularity. It can not be emphatically stated that an individual approach to recommendations would be better than a stereotypical. Again, this depends on the users of our service. A highly heterogenous user base would certainly be better served through individual user models, while a user base that lends itself well to broad segments would perhaps be better served by a type-based approach.

It is comforting to note that most of the papers in our survey report promising results. The increases in *precision* and *recall* range from the negligible to the dramatic. It seems clear then that, since improving an application on these metrics means the user is more satisfied with the presented information, that recommender systems are useful tools for mitigating information overload.

All in all, individual adaption of content would be a welcome addition to an application such as Signal. However, only by collecting extensive knowledge of what constitutes user value in this specific context, and by acknowledging the tradeoff between gathering and using data, can a proper user modeling method be selected.

5.3 Future Work

With all the research on the topic of user modeling, it is interesting to identify possible areas of future work. This section reviews a few possible directions where the current field has yet to explore.

Distributed & Centralized Recommenders As mentioned, most recommender systems are strictly tied to a specific collection of users and items. As described by Ziegler (2005b), an

interesting direction of user modeling is that towards the distributed. In this case, a distributed recommender system would mean a system that leverages information found across the net. Information about items, users and even the system itself gains its strength from a lack of centralization, through increase amounts of data and computing power. Distributed recommender systems can for example be envisioned as a system utilizing a possibly emerging *semantic web* (Ziegler, 2005a).

On the other hand, opposing research in centralized recommendation systems is an interesting field as well. Traditionally, centralized recommenders spanning multiple websites and domains have been the purvey of advertisers. As mentioned, at least one company is developing a centralized, cross-domain and cross-service recommendation system[1]. How this may strengthen estimations and how to best integrate such services with existing infrastructure is an indeed an interesting topic.

[1] See the company Directed Edge at their website, directededge.com

Bridging the Recommender & The Interface While there are many methods for estimating unknown ratings, the data and interface of these methods are remarkably similar. Almost every explored method takes a set of users, items and ratings as input, and provides estimated scalar values as ratings for unknown items through some interface. As the nature of the underlying interface is this homogenous, developing a unifying layer between the recommender and the user interface would be an interesting research topic.

For instance, when considering using personalized recommendations across the content of a website, being able to present this in a more helpful way than a flat, sorted list would be desirable. Accordingly, a layer that transforms simple sorted sets of items into more complex and telling layouts would be an interesting topic for future research. This layer would have to have a host of methods for turning a simple sorted list, along with the data used to compute the list, into visually rich and valuable representations.

Shared Evaluation Metric A common theme across modeling methods examined in this paper is the fact that the comparison of different approaches seems difficult. Each described method often use their own dataset and metrics for measuring the quality of estimated ratings. Future research should be performed into creating a common metric, so that each method can be objectively compared with each other. Naturally, such a metric would have to consider the actual quality of the recommendations through measures such as precision and recall, as well as the computational complexity in relation to different numbers of users and items.

Sub-symbolic methods of Estimation We have seen a few approaches utilizing biologically inspired sub-symbolic AI methods to estimate ratings. It would be interesting to see more research on the effect of using a symbolic versus a sub-symbolic representation of the underlying data when performing rating estimations. As we have seen, the estimation problem relates well to a host of different methods from machine learning, and evaluating even more of these methods would be an interesting possibility for future research.

The Domain-Specific Value of Recommendations On a less technical and more user-centric level, we have the question of which domains would actually benefit from extensive use of recommendations. While the positive answer to this question is taken for granted in a lot of research, it would be interesting to investigate recommendations based on the domain in question. For instance, how well would extensive recommendations work for an online news paper, and which aspects of the implementation and presentation would be crucial? As we have seen in this paper, an application as simple as a feed reader, where the content is presented in flat, sorted lists raise a number of issues when it comes to presentation. Accordingly, organizing the use and customization of recommender systems into domains could be beneficial.

The Usability of Autonomous Web Interfaces Finally, while we did briefly touch on the subject of inherent challenges in the usability of autonomous interfaces, it would be interesting to further research this topic in an online context. While there are obvious benefits to a system that caters to personal needs, the issues of privacy, transparency, accuracy and estimation of interests remain controversial topics. Only by fully understanding these and communicating the flaws and virtues in a way understandable to the average user, can a web application properly use personalized recommendations.

5.4 Conclusions

As demonstrated by the wide variety of different approaches, user modeling is a vast topic with many exciting directions. Factor in new considerations rising from deploying them on the Web, and it becomes clear that the resulting system will be a very complex conception of different parts.

As we have seen throughout this paper, the Web is an excellent field for using recommender systems, and in particular when addressing the usability of content aggregating web applications. Their inherent property of causing information overload makes personalization of content a worth while effort in this domain.

References

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17:734–749.

Al-Shamri, M. Y. H. and Bharadwaj, K. K. (2008). Fuzzy-genetic approach to recommender systems based on a novel hybrid user model. *Expert Systems with Applications*, 35(3):1386 – 1399.

Albert, R. and Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1):47–97.

Albert, R. and Jeong, H. (1999). The diameter of the world wide web. *Arxiv preprint cond-mat/9907038*, pages 1–5.

Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Balabanović, M. and Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Commun. ACM*, 40:66–72.

Barabasi, A.-L., Crandall, A. R. E., and Reviewer (2003). Linked: The new science of networks. *American Journal of Physics*, 71(4):409–410.

Basu, C., Hirsh, H., and Cohen, W. W. (1998). Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *AAAI/IAAI*, pages 714–720.

Bawden, D. and Robinson, L. (2009). The dark side of information: overload, anxiety and other paradoxes and pathologies. *J. Inf. Sci.*, 35(2):180–191.

Billsus, D. and Pazzani, M. J. (1998). Learning Collaborative Information Filters. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46–54, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bogers, T. and van den Bosch, A. (2009). Collaborative and content-based filtering for item recommendation on social bookmarking websites. In *Technical Reports in Computer Science: Recommender Systems the Social Web*. ILK / Tilburg centre for Creative Computing.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370.

Burke, R. (2007). The adaptive web. chapter Hybrid web recommender systems, pages 377–408. Springer-Verlag, Berlin, Heidelberg.

Cato, J. (2001). *User-Centered Web Design*. Pearson Education Limited.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M. (1999). Combining Content-Based and Collaborative Filters in an Online Newspaper.

Dabbish, L. A. and Kraut, R. E. (2006). Email overload at work: an analysis of factors associated with email strain. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 431–440, New York, NY, USA. ACM.

Debnath, S., Ganguly, N., and Mitra, P. (2008). Feature weighting in content based recommendation system using social network analysis. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 1041–1042, New York, NY, USA. ACM.

E. Horvitz, C. M. Kadie, T. P. D. H. (2003). Models of attention in computing and communications: From principles to applications. *Communications of the ACM*, 46:52–59.

Edmunds, A. and Morris, A. (2000). The problem of information overload in business organi-

sations: a review of the literature. *International Journal of Information Management*, 20(1):17 – 28.

Eppler, M. J. and Mengis, J. (2004). The concept of information overload: A review of literature from organization science, accounting, marketing, mis, and related disciplines. *The Information Society*, 20:325–344.

Fischer, G. (2001). User modeling in human–computer interaction. *User Modeling and User-Adapted Interaction*, 11:65–86.

Floreano, D. and Mattiussi, C. (2008). *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. The MIT Press.

Greenberger, M. (1971). *Computers, communications, and the public interest, Volume 1*. The Johns Hopkins Press.

Gregorio, J. and de Hora, B. (2007). The Atom Publishing Protocol. RFC 5023 (Proposed Standard).

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22:5–53.

Hotho, A., Jäschke, R., Schmitz, C., and Stumme, G. (2006). Information retrieval in folksonomies: Search and ranking. In Sure, Y. and Domingue, J., editors, *The Semantic Web: Research and Applications*, volume 4011 of *Lecture Notes in Computer Science*, pages 411–426. Springer Berlin / Heidelberg.

Huang, C.-Y., Sun, C.-T., and Lin, H.-C. (2005). Influence of local information on social simulations in small-world network models. *Journal of Artificial Societies and Social Simulation*, 8(4):8.

Huang, Z., Chung, W., Ong, T.-H., and Chen, H. (2002). A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, JCDL '02, pages 65–73, New York, NY, USA. ACM.

Jameson, A. (2009). Adaptive interfaces and agents. *Human-Computer Interaction: Design Issues, Solutions*, pages 305–330.

Kirsh, D. (2000). A few thoughts on cognitive overload. *Intellectica*, 1(30):19–51.

Kobsa, A. (2001a). Generic user modeling systems. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, 11(1-2):49–63.

Kobsa, A. (2001b). Preface. *User Modeling And User-Adapted Interaction*, 11.

Konstas, I., Stathopoulos, V., and Jose, J. M. (2009). On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 195–202, New York, NY, USA. ACM.

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann.

Lieberman, H. (2009). User Interface Goals, AI opportunities. *AI Magazine*.

Magnini, B. and Strapparava, C. (2001). Improving user modelling with content-based techniques. In Bauer, M., Gmytrasiewicz, P., and Vassileva, J., editors, *User Modeling 2001*, volume 2109 of *Lecture Notes in Computer Science*, pages 74–83. Springer Berlin/Heidelberg.

Mirza, B. J., Keller, B. J., and Ramakrishnan, N. (2003). Studying recommendation algorithms by graph analysis. *Journal of Intelligent Information Systems*, 20:131–160.

Newman, M. E. J., Moore, C., and Watts, D. J. (2000). Mean-field solution of the small-world network model. *Phys. Rev. Lett.*, 84(14):3201–3204.

Norman, D. A. (1998). *The Design of Everyday Things*. The MIT Press.

Nottingham, M. and Sayre, R. (2005). The Atom Syndication Format. RFC 4287 (Proposed Standard).

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web.

Pazzani, M. and Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331.

Pazzani, M. J. and Billsus, D. (2007). The adaptive web. chapter Content-based recommendation systems, pages 325–341. Berlin, Heidelberg.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, pages 175–186, New York, NY, USA. ACM.

Rich, E. (1979). User modeling via stereotypes. *Cognitive Science*, 3(4):329 – 354.

Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). The adaptive web. chapter Collaborative filtering recommender systems, pages 291–324. Springer-Verlag, Berlin, Heidelberg.

Shardanand, U. and Maes, P. (1995). Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, pages 210–217, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Shneiderman, B. (1997). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Smyth, B. (2007). Case-based recommendation. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 342–376. Springer Berlin / Heidelberg.

Thomas H. Davenport, J. C. B. (2001). *The attention economy: understanding the new currency of business*. Harvard Business School Press.

Totterdell, P. and Rautenbach, P. (1990). Adaption as a problem of design. pages 59–84.

Ujjin, S. and Bentley, P. (2002). Learning user preferences using evolution. In *The 4th Asia-Pacific Conference on Simulated Evolution and Learning*.

Ujjin, S. and Bentley, P. (2003). Particle swarm optimization recommender system. In *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, pages 124–131.

Uwe Malinowski, Thomas Kühme, H. D. and Schneider-Hufschimdt, M. (1992). A taxonomy of adaptive user interfaces. pages 391–414.

Walter, F., Battiston, S., and Schweitzer, F. (2008). A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16:57–74.

Webb, G. I., Pazzani, M. J., and Billsus, D. (2001). Machine learning for user modeling. *User Modeling and User-Adapted Interaction*, 11:19–29.

Ziegler, C.-N. (2005a). Semantic web recommender systems. In Lindner, W., Mesiti, M., Türker, C., Tzitzikas, Y., and Vakali, A., editors, *Current Trends in Database Technology - EDBT 2004 Workshops*, volume 3268 of *Lecture Notes in Computer Science*, pages 521–521. Springer Berlin / Heidelberg.

Ziegler, C.-N. (2005b). *Towards Decentralized Recommender Systems*. PhD thesis, Albert-Ludwigs-Universität Freiburg.

Zukerman, I. and Albrecht, D. W. (2001). Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, 11(1-2):5–18.