ASSIGNMENT

User Meta Modeling

Aggregating User Modeling Methods
On a Personal Level

Study how to create composite user models by combining results from numerous complimenting modeling algorithms. Create a flexible algorithm that individually combines the results into one coherent user model. Utilize the resulting composite user modeling algorithm in an information retrieval system to provide personalized search.

Contents

— 1 —
Introduction

In 1971, Herbert Simon said the following on the topic of information overload: "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." Greenberger (1971).

In 2009, Alon Halevy, Peter Norvig, and Fernando Pereira, wrote: "Perhaps when it comes to natural language processing and related fields, we're doomed to complex theories that will never have the elegance of physics equations. But if that's so, we should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data." Halevy and Norvig (2009).

Previously, something else was the problem

Today, biggest problems on the web

Information overload

Content discovery

Search

User modeling

Often generic methods

the modeling problem: model+prediction

the core problem: estimating preferences

getting past 80%

the efficiency of data

This paper: A more personal approach

Aggregated user modeling methods for truly personal predictions.

Hypothesis

Contributions

Outline

— 2 —

Theory

This chapter will introduce some basic theory needed to develop our approach to user modeling. We will first describe our stated enemy, the information overload problem, before delving into how user modeling and, more specifically, recommender systems, is currently used to solve this problem.

This chapter will also introduce the notion of personalized search, a field where our user modeling method will be especially applicable. The next chapter will use these theories to build an *even more personalized approach* to user modeling.

2.1 Information Overload

Information overload conveys the act of receiving *too much information*. The problem is apparent in situations where decisional accuracy turns from improving with more information, to being hindered by too much irrelevant data (Bjorkoy, 2010, p13). Needness to say, this 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).

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.

Two common tasks quickly become difficult in this situation: Consuming content that is known by the user to be relevant can be drowned out by irrelevant noise. Orthogonally, discovering new, interesting yet unknown content also becomes difficult because of the sheer amount of available content. Finding contemporary examples is not difficult:

- Missing important news articles that get drowned out by irrelevant content.
- Forgetting to reply to an email as new messages keep arriving.
- Discovering sub-par movies because those most relevant are never discovered.

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. In other words, to solve the problems of information overload and content discovery, applications must be able to individually adapt to each user.

An insightful perspective on information overload comes from the study of attention economy. In this context human attention is seen a scarce commodity, offset by how much irrelevant noise is present at any given time. 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" (Davenport and Beck, 2001). To evade information overload is then to maximize available attention, allowing more focus on the most important items of an interface.

Conceptual models used in interaction design can help us see when and where information overload interferes with the user experience. Norman (1988) advocates a model called the seven stages of action, describing how each user goes through several states while using a system (see Figure 2.1). 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. 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.

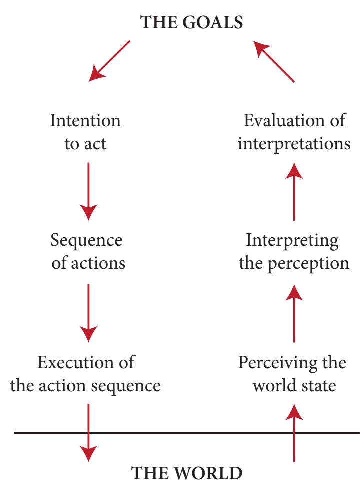

Figure 2.1

As apparent from this model, 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. Information overload happens throughout the interaction process, which is important to know when considering possible solutions.

Online Overload

The Web is a common source of information overload. As we will use the web as an example throughout this paper, this section describes why the Web is so conducive to information overload.

Online information overload is especially pervasive when considering *content aggregating websites*, i.e. sites that collate information from multiple other sites and sources. Online information retrieval (i.e. search engines), fall into this category, as does online newspapers, feed readers and portal websites. As mentioned, the wealth and scope of data are natural culprits of online overload, as well as the varying qualities of websites publishing the information. However, lessons from graph theory can also help us see why information overload occurs on the Web.

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. These models also show a theoretical foundation for why information overload occurs. In the Web graph, 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 and Moore, 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 2.2). 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. As described by Barabási (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

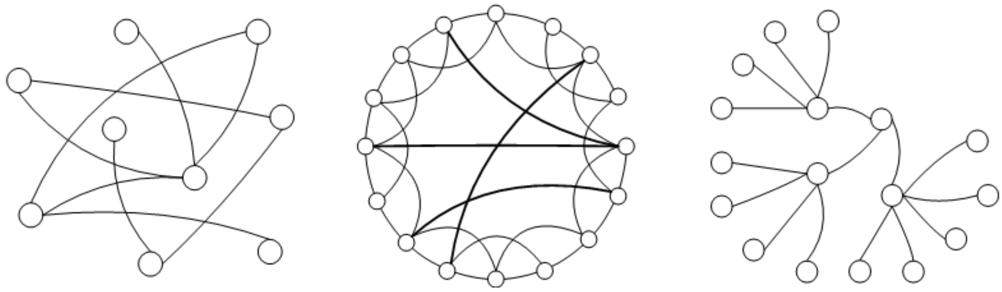

Figure 2.2: **Complex Networks:** From the left: A *random* network, a *small-world* network and a *scale-free* network (which is a type of small-world network). Figure adapted from Huang et al. (2005).

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 et al. (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 (Barabási, 2003, p71).

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 two power laws: $P_{in}(k) \sim k^{-\gamma_{in}}$ and $P_{out}(k) \sim k^{-\gamma_{out}}$.

Albert et al. (1999) 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 (or long tails). In other words, a few important nodes have a huge number of inbound and outbound links — the hubs. Barabási (2003, p86) proposed that hubs emerge in a scale-free networks because of two factors: (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 et al., 1999), and the probability for a new node connecting to an existing node is given by $\prod k_i$ in Equation 2.1, where k_i is the number of links pointing to node i.

$$\prod_i k_i = \frac{k_i}{\sum_j^N k_j} \tag{2.1}$$

Another important topological observation of the Web is how fragmented it is. Barabási (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. 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.

So far we have established that information overload is a pervasive problem, especially on the web. The question now becomes how to best solve this issue. This is where user modeling comes in.

2.2 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.

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 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.

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. Examples of such user modeling include:

- Translating content based on a user's geographical location.
- Suggesting interesting items based on previous activity.
- Reorganizing or filtering content based on predicted user relevance.
- Changing the presentation of content to match personal preferences or abilities.

The fields of Artificial Intelligence (AI) and Human-Computer Interaction (HCI) share a common goal solving information overload through user modeling. However, as described by Lieberman (2009), they have different approaches and 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. Luckily, according to Kobsa (2001), recent research has blurred the lines between the AI and HCI in user modeling.

In AI, user modeling refers to precise algorithms and methods that infer knowledge about a user based on past interaction (e.g. Pazzani and Billsus (2007); Smyth (2007); Alshamri and Bharadwaj (2008); Resnick et al. (1994)). 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. For instance, an individual user model may predict how interesting 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 (e.g. 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. 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."

This paper is concerned with the AI approach to user modeling, and in particular, the use of *recommender systems (RS)*. As our goal is to combine different RSs into one coherent user model, we will now describe what an RS entails, and introduce some of the many algorithms they employ.

2.3 *Recommender Systems*

The name might seem constraining, but recommender systems are incredibly powerful methods in user modeling. Whenever we wish to predict the relevance of an item to a user, recommender systems are the tools to use. Such systems are commonly used on the web to provide a host of predictive functionality, including:

- Recommending products like books or movies based on past purchases.
- Suggesting new social connections based on an existing social graph.
- Recommending items based the activity of similar or like-minded users.
- Ordering news articles by predicted individual relevance.
- Personalizing search results based on the current user.

Common to these examples are a set of users, a set of items, and a sparse set of explicit ratings or preferences. Items can be anything: Documents, movies, music, places, people, or indeed other users. A recommender system is best described by graph and graph operations, even though the underlying algorithms might not use this as the representation. Mirza and Keller (2003) explains how any RS can be expressed as a graph traversal algorithm. Items and users are nodes, while ratings, social connections et cetera are edges between the nodes. An RS performs predictive reasoning on this graph by estimating the strenghts of hypothetical connections between nodes that are not explicitly connected.

For example, if a user has rated some of the movies in a movie recommendation system, we use these ratings to predict how well the user will like unseen movies, based on a movies ratings from users similar to the one in question. In social networks, recommender systems can be used to infer new social relations based on existing connections. The principle is the same: By evaluating current explicit connections, and the connections of similar users, new connections can be predicted. Recommender systems are then powerful methods for user modeling, personalization and fighting information overload, because of their ability to infer how relevant and item (or another user) will be to the current user.

Formally, a recommender system can be seen as a quintuple, $\mathrm{RS} = (I, U, R, F, M)$, where I is the set of items (e.g. products, articles or movies) and U is the set of users. R is the set of known connections, for example explicit preferences given by users for certain items, or connections in a social graph. F is a framework for representing the items, users and ratings, for example a graph or matrix. M is the actual user modeling method used to infer unknown ratings for predicting a user's preference for an unrated item. This is where AI comes in.

In Adomavicius and Tuzhilin (2005), M is seen as a utility function $f : U \times I \to S$. Here, f is a function that maps the set of users and items into a fully ordered set of items S, ranked by their utility (i.e. rating) to each user. In other words, S is the completely specified version of R, where each user has either an explicit, implicit or predicted preference for each item in I. To predict the best unrated item for each user, we simply find the item with the highest expected utility:

$$\forall u \in U,\ i'_u = \arg\max_{i \in I} f(u, i)$$

The utility function u depends on the modeling method being used, the active user and the item in question. The *reason* for using a recommender system is that the utility u is not defined for the entire $U \times I$ 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.

Another popular way of describing, and implementing an RS is using a simple matrix. Here, one dimension represents users, the other dimension represents items, and each cell corresponds to an explicit rating. This matrix then becomes the framework F in our RS quintuple:

$$R_{u,i} = \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,i} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,i} \\ \vdots & \vdots & \ddots & \vdots \\ r_{u,1} & r_{u,2} & \cdots & r_{u,i} \end{pmatrix}$$

Critically, these matrices are usually extremely sparse (i.e. most of the cells are empty). Consider that while there may be a large number of users and items, each individual user only rates or connects to a few number of items. For example, in the seminal Netflix Challenge movie recommender dataset, almost 99% of the potential user/item pairs have no rating (Bell and Koren, 2007, p1). In other words, the recommender system must be able to produce results from a matrix where only 1% of the cells have meaningful values.

Naturally, this is the defining characteristic of many recommender systems: the ability to extract meaningful patterns from sparse data, through dimensionality reduction, neighborhood estimation and other methods, as we shall see.

Recommender systems face many challenges other than the sparsity problem. A directly related problem is the need for large datasets. Since the data is often sparse, the systems will most often perform well if used on large numbers of items and users. As in many machine learning methods, concept drift, where the characteristics of a user or item changes over time, is also always present.

The performance of RSs is often closely tied to their computational complexity. Real world usage of the most precise methods is often hindered by the computational power needed to actually put them into production.

Finally, the scale of the data in question should be a concern. If the ratings are ordinal data (e.g. 1-5) input directly by users, the RS should take into account the domain specific meaning of these intervals. For example, in a system for rating movies, the jump between ratings 4-5 might not have the same significance as the jump from 2-3. However, this is a fact seldom mentioned in the literature. Most RSs employ metrics that assume a normal distribution, and even the common evaluation techniques such as RMSE or MAE treat ordinal data as a continous scale.

Prediction

The most interesting and important part of any RS is how it predicts unknown ratings. (Note that altough we use "ratings", "utility", "preference", "relevance" and "connection strength" depending on the context, they all basically mean the same.) Because of this, each method is best categorized based on a few dimensions of its predictive capabilities (see Table 2.1). In our taxonomy, these dimensions are: *data*, *method*, *granularity*, *temporality* and *agents*.

The *data* variable represents what data the RS uses to perform predictions. Content-based methods use only the items, inter-item relations, and 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 personal models can be created. However, such methods often require a lot of interaction before reliable models can be created (Adomavicius and Tuzhilin, 2005). The problem of having to do complex inference from little data, as is often is in content-based predictions, 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. When using content-based predictions, the utility function $f(u,i)$ of user u and item i is extrapolated from $f(u,i_u)$, where i is an item similar to i_u and $f(u,i_u)$ is known.

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

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

Table 2.1: A taxonomy of recommender systems. From Bjorkoy (2010).

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 prediction. Collaborative models must be careful not to represent the *average* user, but a single individual. When using a collaborative method, the utility $f(u, i)$ of item i for user u is extrapolated from $f(u_j, i)$ where u_j is a user similar to u.

Because of *the new user problem* of content-based prediction and the *average user problem* of collaborative prediction, many systems use a hybrid approach (Burke, 2007). By combining content-based and collaborative methods, systems that properly handle predictions for new users and avoid too much generalization in the models can be achieved.

The *method* variable, is another way to classify recommenders. Orthogonal to what data the method uses, this variable concerns *how* the data is used to produce recommendations. 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.

The *granularity* variable tells whether this approach creates models for the canonical user, stereotypical users or individual users. Rich (1979) presented one of the first user modeling systems based on stereotypes, used to predict which books in a library each user would most enjoy. Here, a dialogue between the system and the user was performed to place the user into a set of sterotypes. Each stereotype has a set of *facets* which is then used to match books and users.

Temporality refers to how volatile the gathered knowledge will be. While most RSs produce long term, relatively stable knowledge based on lasting user preference and taste, some systems use fluctuating parameters such as the time of day, exact location and the current context to produce recommendations. For example, Horvitz et al. (2003) used clues from a user's calendar, camera and other sensors to determine the attentional state of the user before delivering personalized and contextual notifications.

The *agents* variable signifies whether the knowledge gathering and presentation is implicit and opaque, or explicit and requires dedicated user interaction. Explicit feedback through ratings is common in movie, product or music rating services (e.g. Bell et al. (2007); Basu et al. (1998); Hotho et al. (2006)). However, for other services such as personalized search, implicit mining of query logs and user interaction is often used to build user models (e.g. Shen et al. (2005); Agichtein et al. (2006); Speretta and Gauch (2000); Teevan et al. (2005))

Approaches

Because our solution will combine different recommender systems, we need a short introduction to some of the approaches we will combine. See Adomavicius and Tuzhilin (2005), Pazzani and Billsus (2007), Schafer et al. (2007) or Bjorkoy (2010) for a more comprehensive exploration of different types of recommenders.

Baseline ratings are the simplest family of recommender systems, based on item and user rating averages. The data is content-based, and used to compute heuristic predictions. This is done on a per-user, individual basis and collects long term knowledge so far as the user is rational in most of his or her ratings. While simple in nature, they are often helpful as starting points for more complex systems, or as benchmarks for exploring new approaches. (Koren, 2008, p2) computes the baselines for items and users, and use more involved methods to move this starting point in some direction. The baseline for a user/item pair is given by

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

where μ is the average system rating, b_u is the user baseline and b_i is the item baseline. The user and item baselines correspond to how the user's and item's ratings deviate from the norm. This makes sense as some items may be consistently rated higher than the average, some users may be highly critical in their assesments, and so on. Koren computes these baselines by solving the least squares problem

$$\min_{b*} = \sum_{(u,i)\in R} (r_{ui} - \mu - b_u - b_i)^2 + \lambda(\sum_u b_u^2 + \sum_i b_i^2)$$

which finds baselines that fit the given ratings while trying to reduce overfitting (as weighted by the λ parameter). By using baselines instead of simple averages, more complex predictors gain a better starting point, or in other words, a better average.

Another approach based on simple averages is the *Slope One* family of collaborative filtering algorithms. As introduced by Lemire and Maclachlan (2005), these algorithms predict unknown ratings based on the average difference in ratings between two items. For example, if item i is on average rated δ points above item j, and the user u has rated item j, that is, we know r_{uj}, the predicted rating of i is simply $r_{uj} + \delta$. While simplistic, Slope One is computationally effective and produces results comparable to more complex methods (Lemire and Maclachlan, 2005, p5).

Dimensionaliy reduction is a technique often used by recommendation systems.

Neighborhoods knn, pearson

IR methods vector space, page rank

Social networks traversal, transitive trust

2.4 *Personalized Search*

Information retrieval (+ information overload)

An Information Retrieval Model is a quadruple (Baeza-Yates and Ribeiro-Neto, 1999, p23):

$$\mathrm{IR} = (Documents, Queries, Framework, ranking(q_i, d_i)) \tag{2.2}$$

Common metrics

Personalized metrics

Relation to recommender systems

2.5 *Aggregate Modeling*

Current methods (non-personal)

Use cases (netflix, ...)

For personalized search (speculation)

Explain next chapter

— 3 —
Methods

Getting past 80%

Power of data

Hypotheses

3.1 *Modeling Phase*

$$\mathrm{AM} = (Items, Users, Framework, Methods, Aggregation) \tag{3.1}$$

3.2 *Prediction Phase*

3.3 *Implementation*

3.4 *Evaluation*

Datasets

Metrics

Experiments

— 4 —
Results

4.1 Experiments

4.2 Results

4.3 Discussion

— 5 —
Conclusion

5.1 Contributions

Findings

Answering Hypotheses

5.2 Future Work

5.3 Conclusion

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(6):734–749.

Agichtein, E., Brill, E., Dumais, S., and Ragno, R. (2006). Learning user interaction models for predicting web search result preferences. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '06*, page 3.

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

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

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern information retrieval*, volume 463. ACM press New York.

Barabási, A. (2003). Linked: The new science of networks. *American journal of Physics*.

Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 714–720. JOHN WILEY & SONS LTD.

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

Bell, R., Koren, Y., and Volinsky, C. (2007). The BellKor solution to the Netflix prize. *KorBell Team's Report to Netflix*.

Bell, R. M. and Koren, Y. (2007). Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75.

Bjorkoy, O. (2010). User Modeling on The Web: An Exploratory Review.

Burke, R. (2007). Hybrid Web Recommender Systems.

Cato, J. (2001). *User-centered web design*. Pearson Education.

Davenport, T. and Beck, J. (2001). *The attention economy: Understanding the new currency of business*. Harvard Business Press.

Edmunds, A. and Morris, A. (2000). The problem of information overload in business organisations: a review of the literature. *International Journal of Information Management*, 20(1):17–28.

Eppler, M. 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(5):325–344.

Fischer, G. (2001). User modeling in human–computer interaction. *User modeling and user-adapted interaction*, 11(1):65–86.

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

Halevy, A. and Norvig, P. (2009). The unreasonable effectiveness of data. *Intelligent Systems, IEEE*, 24(2):8–12.

Horvitz, E., Kadie, C., Paek, T., and Hovel, D. (2003). Models of attention in computing and communication: from principles to applications. *Communications of the ACM*, 46(3):52–59.

Hotho, A., J, R., Schmitz, C., and Stumme, G. (2006). Information Retrieval in Folksonomies: Search and Ranking.

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

Jameson, A. (2009). Adaptive interfaces and agents. *Human-Computer Interaction: Design Issues, Solutions, and Applications*, page 105.

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

Kobsa, A. (2001). Generic User Modeling Systems.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM.

Lemire, D. and Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering. *Society for Industrial Mathematics*.

Lieberman, H. (2009). User interface goals, AI opportunities. *AI Magazine*, 30(2).

Mirza, B. and Keller, B. (2003). Studying recommendation algorithms by graph analysis. *Journal of Intelligent Information*.

Newman, M. and Moore, C. (2000). Mean-field solution of the small-world network model. *Physical Review Letters*.

Norman, D. (1988). The design of everyday things. *Basic Book Inc./New York*.

Pazzani, M. and Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer-Verlag.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens : An Open Architecture for Collaborative Filtering of Netnews.

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

Schafer, J., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. *The adaptive web*, pages 291–324.

Shen, X., Tan, B., and Zhai, C. (2005). Implicit user modeling for personalized search. *Proceedings of the 14th ACM international conference on Information and knowledge management - CIKM '05*, page 824.

Smyth, B. (2007). Case-Based Recommendation.

Speretta, M. and Gauch, S. (2000). Personalized Search Based on User Search Histories. *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pages 622–628.

Teevan, J., Dumais, S. T., and Horvitz, E. (2005). Personalizing search via automated analysis of interests and activities. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '05*, page 449.

Totterdell, P. and Rautenbach, P. (1990). *Adaption as a Problem of Design*, pages 59—-84.

Appendix

Sourcecode, tables, et cetera.