Aalto University

School of Science

Department of Mathematics and Systems Analysis

Jesse Myrberg

A Recommender System for an Online Auction

Espoo, 2.8.2016

Supervisor: Prof. Ahti Salo

Advisor: M.Sc. (Tech.) Hugo Gävert

Author Jesse Myrberg

Title of thesis A Recommender System for an Online Auction

Degree programme Master's Programme in Mathematics and Operations Research

Major Systems and Operations Research **Code of major** SCI3055

Thesis supervisor Prof. Ahti Salo

Thesis advisor(s) M.Sc. (Tech.) Hugo Gävert

Date 2.8.2016 **Number of pages** vii + 85 **Language** English

The development of recommender systems arise from the observation that individuals often rely on recommendations provided by others. Recently, new electronic business services with massive amount of information for users to filter, have caused a pressing need for recommender systems. Yet, displaying irrelevant items on electronic commerce sites may drive away potential customers and lead to loss of business opportunities. In this thesis, a recommender system is built for a major Finnish online auction site.

This thesis aims to build a recommender system that makes the front page of the auction site more relevant and personalized for the users of the site. This is achieved by allowing the recommender system to generate item recommendations to display on the front page. In order to find a suitable recommender system for the auction site, a total of seven different recommender systems are built and compared. The evaluation is based on predictive accuracy of the recommenders, and it is done in an offline simulation using the browsing history of the auction site users.

Most of the literature on recommender systems assume that items to be recommended are static and non-unique. However, in this thesis, recommendations need to be made in a dynamic environment where items are short-lived and unique. The problem is approached by creating a general recommendation process that can be used as a basis for content-based recommenders that use natural language processing methods for representing auction items as vectors.

The evaluation results show that good predictive accuracy can be achieved with recommenders that are based on totally different approaches. The neural-network based Word2Vector (W2V) model and more traditional Term Frequency-Inverse Term Frequency (TFIDF) based models work well on recommending new items to the users. On the other hand, the results suggest that users tend to revisit items they have previously visited, and offering a shortcut to previously visited items on the front page might be reasonable.

Keywords recommender system, natural language processing, online auction

Tekijä Jesse Myrberg	
Työn nimi Suositusjärjestelmä nettihuutokaupalle	
Koulutusohjelma Matematiikan ja operaatiotutkimuksen maisteriohjelma	
Pääaine Systeemi- ja operaatiotutkimus	**Pääaineen koodi** SCI3055
Työn valvoja Prof. Ahti Salo	
Työn ohjaaja(t) DI Hugo Gävert	
Päivämäärä 2.8.2016	**Sivumäärä** vii + 85 — **Kieli** Englanti

Yksilöt usein turvautuvat päätöksenteossaan toisten antamiin suosituksiin, oli kyse sitten esimerkiksi musiikista, kirjoista tai elokuvista. Suosittelujärjestelmien kehitys on peräisin tästä havainnosta. Viime vuosina sähköisen liiketoiminnan kehityksen ja kasvavan informaatiomassan myötä suositusjärjestelmille on syntynyt kasvava tarve. Irrelevantit ja personoimattomat elektronisen liiketoiminnan sivustot voivat pahimmillaan ajaa pois potentiaalisia asiakkaita ja johtaa taloudellisiin tappioihin liiketoiminnassa. Tässä työssä rakennetaan personointia edistävä suositusjärjestelmä suurelle suomalaiselle nettihuutokaupalle.

Työssä on tavoitteena rakentaa suositusjärjestelmä, joka tekee nettihuutokaupan etusivusta relevantimman ja personoidumman sen käyttäjille. Tämä saavutetaan suositusjärjestelmällä, joka tekee käyttäjän selaushistorian perusteella huutokaupan etusivulle personoituja huutokauppakohdesuosituksia. Sopivan suositusjärjestelmän löytämiseksi työssä vertaillaan ja evaluoidaan yhteensä seitsemän eri suositusjärjestelmää. Evaluointi perustuu suositusjärjestelmien ennustustarkkuuteen, ja se toteutetaan offline-simulaatiolla hyödyntäen käyttäjien selaushistoriaa.

Suuri osa suositusjärjestelmiin liittyvästä kirjallisuudesta olettaa suositeltavien kohteiden tai esineiden olevan staattisia ja ei-uniikkeja. Näin ei kuitenkaan ole huutokaupassa, minkä vuoksi tässä työssä suositusjärjestelmän tulee kyetä suosittelemaan lyhytikäisiä ja uniikkeja kohteita. Työssä ongelmaa lähestytään luomalla yleinen suositusprosessi, joka toimii pohjana sisältöperusteisille suositusjärjestelmille, jotka hyödyntävät luonnollisen kielen prosessoinnin menetelmiä huutokauppakohteiden esittämisessä vektorina.

Tulosten perusteella hyvä ennustetarkkuus voidaan saavuttaa erilaisiin lähestymistapoihin perustuvilla suositusjärjestelmillä. Sekä neuroverkkoihin perustuva Word2Vector (W2V) että Term Frequency-Inverse Term Frequency (TFIDF) – mallit soveltuvat ennustetarkkuudeltaan parhaiten käyttäjälle uusien kohteiden suosittelemiseen. Tulosten perusteella käyttäjille voisi olla järkevää suositella myös samoja kohteita, joissa he ovat aiemmin käyneet, ja näin ollen tehdä etusivusta käyttäjäystävällisempi tarjoamalla suora oikotie aiempiin kohteisiin.

Avainsanat suositusjärjestelmä, luonnollisen kielen prosessointi, nettihuutokauppa

Acknowledgements

First, I want to thank my thesis instructor Hugo for providing me this interesting thesis opportunity and allowing me to do it independently with a flexible schedule. As a data enthusiast, it has truly been a unique opportunity to be able to explore such massive and interesting datasets. I have definitely learned a lot during this process.

Second, I thank my supervisor Professor Ahti Salo for his guidance on this thesis, and also on my Bachelor's thesis. I also want to express my thanks to the Department of Mathematics and Systems Analysis for the interesting courses and guidance.

Finishing the last few courses and this thesis alongside working as a part-time IT consultant has not always been easy. I want to thank my family for the continuous encouragement and support to finish this thesis. Especially, I want to thank Salla, and the two little tail waggers Taffy and Cora for making me smile every day.

Espoo, 2.8.2016

Jesse Myrberg

Abbreviations

DCG	Discounted Cumulative Gain
EM	Expectation Maximization
FN	False-Negative
FP	False-Positive
LDA	Latent Dirichlet Allocation
LDAub	User-based Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
LSI	Latent Semantic Indexing
LSIub	User-based Latent Semantic Indexing
MAE	Mean Absolute Error
NDCG	Normalized Discounted Cumulative Gain
R20 / R100	R-Score with parameter $\alpha = 20$ / $\alpha = 100$
RMSE	Root Mean Square Error
SISC	Simple Item Space Coverage
SVD	Singular Value Decomposition
TFIDF	Term Frequency—Inverse Document Frequency
TN	True-Negative
TP	True-Positive
TPR	True-Positive-Rate
VSM	Vector Space Model

Contents

1 Introduction

1.1 Background and motivation

When one needs to make a decision with insufficient experience on the alternatives, it is often needed to rely on recommendations from other people. The recommendations can originate from multiple different sources, such as by word of mouth, letters, reviews or general surveys. Recommender systems assist in this social decision-making process by taking recommendations as inputs, then aggregating them, and finally sending them to appropriate recipients. The value of recommender systems comes from both the aggregation and the system's ability to make matches between the recommenders and those who seek recommendations (Resnick & Varian, 1997).

The development of recommender systems arise from the observation that individuals often rely on recommendations provided by others in their decision-making process. For example, individuals tend to read and rely on the movie reviews when they are selecting a movie to watch. Also, employers tend to count on recommendation letters when recruiting new employees (Ricci, et al., 2011; Schwartz, 2004). In addition to filtering undesired alternatives, recommender systems can suggest particularly interesting alternatives to its users (Gündüz-Ögüdücü, 2010).

Recently, new electronic business services have caused a pressing need for recommender systems. In fact, recommender systems have developed in parallel with the Web. This is due to the fact that there is a vast number of alternatives being offered to users, which makes it difficult for users to choose between alternatives.

Examples of these electronic business services are online shops, online auctions, and different product comparison services (Ricci, et al., 2011; Bobadilla, et al., 2013).

Currently, nearly everyone who surfs the Web sites on Internet will come across recommender systems. There are many popular Web sites that exploit recommender systems, such as *Amazon*[1], *YouTube*[2] and *Netflix*[3] (Gündüz-Ögüdücü, 2010).

1.2 Scope and objectives

This thesis has been developed in collaboration with a major Finnish online auction, where users can buy and sell a variety of different items, such as antique, cars, movies, clothes, electronics and artwork. Users vary from single individuals to large corporate businesses. The auction site has over 500 000 users every week, and about 200 000 € worth of products are sold per day.

Currently, the front page of the auction site displays items that are irrelevant for the users. This may drive away potential users and lead to loss of business opportunities. Moreover, users are required to filter the most interesting and relevant items themselves from a very large and dynamic collection of approximately 1.5 million unique items, which can be a laborious task. Thus, a recommender system is needed to make the site more relevant for the users.

The goal of this thesis is to build a recommender system that makes the front page of the online auction site more relevant for the users. In practice, multiple recommenders are evaluated in an offline simulation, using real life clickstream data of the users.

1.3 Structure of the thesis

This thesis is divided in to six chapters. Chapter 2 introduces the theory of recommender systems and online auctions, and examples of recommender systems in real-life scenarios. Chapter 3 presents the most relevant recommender evaluation

[1] http://www.amazon.com
[2] http://www.youtube.com
[3] http://www.netflix.com

methods in the context of this thesis. Chapter 4 describes the research problem and justifies the techniques and processes that are used in the construction of the recommenders. Moreover, the data, experimental setup and evaluation methods used are explained. Chapter 5 presents the simulation results. Finally, Chapter 6 discusses the developed recommendation systems and summarizes this thesis.

2 Theory and practice

2.1 Recommender systems

2.1.1 Former research

The study of recommender systems is new compared to research into other classical information system tools and techniques, such as databases and search engines (Ricci, et al., 2011). The first research paper on recommender systems was published in the mid-1990s (Park, et al., 2012). The first actual recommender system was Tapestry, an experimental mail system for filtering vast number of incoming documents by utilizing user collaboration (Goldberg, et al., 1992). Tapestry was the first recommender system that utilized *collaborative filtering,* a recommendation technique still used by many highly rated Web sites, such as *Amazon* and *YouTube* (Linden, et al., 2003; Davidson, et al., 2010; Resnick & Varian, 1997).

Since the mid-1990s, interest in recommender systems has dramatically increased. This is supported by the following facts (Ricci, et al., 2011):

- Recommender systems play a major part in popular Internet sites such as Amazon, YouTube, *Netflix*, *Yahoo*[4], *Last.fm*[5] and *IMDB*[6]. For example, in 2006, Netflix announced the Netflix Prize challenge (Amatriain & Basilico, 2012), where $ 1 million was offered to a team that could improve the accuracy of the current recommender system by 10 %.

[4] http://www.yahoo.com
[5] http://www.last.fm
[6] http://www.imdb.com

- Since 2007, ACM Recommender Systems (RecSys) has been an annual event focusing on recommender technology research and applications. Moreover, recommender system topics are covered in many other conferences, such as ACM Special Interest Group on Information Retrieval (SIGIR), User Modeling, Adaptation and Personalization (UMAP), and ACM's Special Interest Group on Management of Data (SIGMOD).
- At higher education institutions, many courses are dedicated to recommender systems only. In addition, tutorials on recommender systems are common in computer science conferences.
- Special issues in academic journals about research and development of recommender systems field, such as AI Communications (2008), IEEE Intelligent Systems (2007) and ACM Transactions on Computer-Human Interaction (2005).

The number of published research papers between years 2001 and 2010 was 210 in total. Majority of these research papers were related to movies (25.2 %) and shopping (20.0 %), because these fields have a large number of practical applications. The most popular journals in recommendation field between years 2001 and 2010 were *Expert Systems with Applications* (33.3 %) and *IEEE Intelligent Systems* (10.0 %). In recent years, the number of research papers has been increasing, as shown in Figure 1 (Park, et al., 2012).

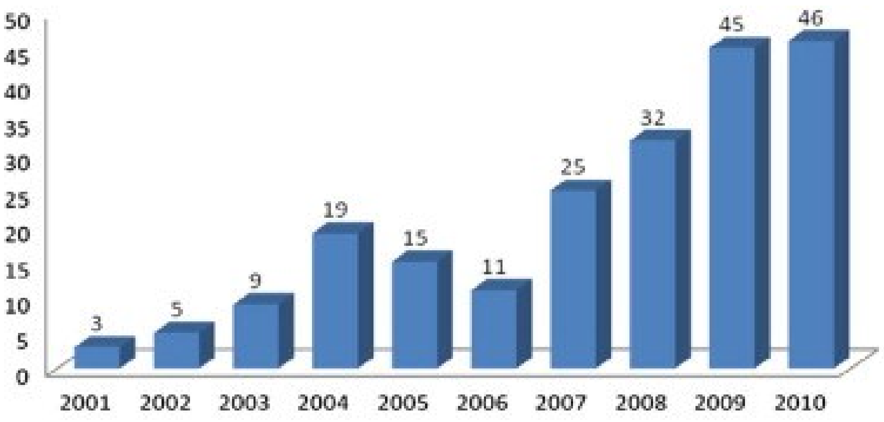


Figure 1: The distribution of research papers by year of publication. The figure is taken from *A literature review and classification of recommender systems research* (Park, et al., 2012).

2.1.2 Core concepts and definitions

Recommendation systems are defined as software tools and techniques that provide suggestions for items to be of use to a user (Burke, 2007). In this thesis, recommender systems are also referred to as *recommender systems* and *recommenders*. The definition of a recommender system includes some core concepts that are explained in more detail in the following (Ricci, et al., 2011; Barbieri, et al., 2014; Burke, 2007).

Recommendation is an information filtering form that analyzes users' historical preferences on a catalog of items in order to generate a personalized list of suggested items. It is an option worthy of consideration.

Users are those who use the recommender system, and to whom the items are recommended. Users may have different goals and characteristics. Recommender systems exploit information about the users in order to personalize the human-computer interaction.

Items denote objects that the system recommends to users, and they can be characterized by their complexity, value or utility. The value of an item is positive, if the item is useful to the user, and negative, if the item is not useful for the user. Recommender systems focus in various different fields, and items can be for example CDs, documents or news. Recommender systems are also customized in such way that the provided recommendations are useful and effective suggestions for a specific item type.

In the following, a recommender system is defined mathematically as a scoring function, as described by Barbieri et al. (2014) in *Probabilistic Approaches to Recommendations* (Barbieri, et al., 2014).

Items, users and their preferences are denoted as follows. Let $\mathcal{U} = \{u_1, \dots, u_M\}$ be a set of M users and $\mathcal{I} = \{i_1, \dots, i_N\}$ a set of N items, where u represents a single user, and i represents a single item. Using this notation, users' preferences can be modeled with a $M \times N$ size matrix $\boldsymbol{R}$, where each element r_i^u of $\boldsymbol{R}$ denotes the preference value that user u assigns to item i. In this thesis, this preference value is referred to as *rating*, which is interpreted as the degree of user's appreciation for a certain item i. Since $\boldsymbol{R}$ describes the users' preferences, it is also referred to as a *rating matrix*.

User preference data (elements r_i^u for each user u and item i combination) can be *explicit* or *implicit*. Explicit user preference data are explicitly expressed by individual users on the items they have experienced. For example, explicit user preference data are retrieved when users rate movies from 1 (worst) to 5 (best) after watching them (thus, $r_i^u \in \{1, 2, 3, 4, 5\}\ \forall u, i$). In this thesis, the focus is on implicit user preference data, which are observations of user and item co-occurrences. Examples of implicit user preference data are users' recorded Web sessions, likes, viewing times, check-ins, and clickstreams on Web pages.

Ratings can be explicit or implicit user preference data. A generic entry of user-item rating matrix $\boldsymbol{R}$ can be defined as a binary value. If user u has not yet experienced (for example, in implicit case not clicked on the item's Web page, or in explicit case not rated the item) item i, then implicit rating $r_i^u = 0$. On the other hand, if user u has experienced item i, then the generic entry for rating matrix is implicit rating $r_i^u = 1$.

Let $\langle u, i \rangle$ be the enumeration of all user-item pairs in $\boldsymbol{R}$. By using the previous notation, the set of items (implicitly or explicitly) rated by user u can be defined as $\mathcal{I}_{\boldsymbol{R}}(u) = \{i \in \mathcal{I} \mid \langle u, i \rangle \in \boldsymbol{R}\}$. Similarly, $\mathcal{U}_{\boldsymbol{R}}(i) = \{u \in \mathcal{U} \mid \langle u, i \rangle \in \boldsymbol{R}\}$ can be defined as the set of users that have (implicitly or explicitly) rated item i. With this notation, it is possible to define the following key concepts:

Active user is any user u that has rated more than or equal to one item, and thus for an active user it satisfies that $\mathcal{I}_{\boldsymbol{R}}(u) \neq \emptyset$. This means that an active user has rating history.

Cold-start problem occurs when no items have been rated by users u, or respectively, no users have rated items i. It commonly occurs when a new user or item is added to the underlying information system, and the recommender system cannot provide suggestions in the absence of information.

As mentioned before, a recommender system provides suggestions for items to be of use to a user. Because this is the case, a recommender system aims to provide an active user u with a list of item recommendations, *recommendation list* $\mathcal{L}_u \subseteq \mathcal{I}$, where the items are expected to be of the user's interest. Often the recommendation list may only include items from which the user u has no experience, and thus, it must satisfy that $\mathcal{L}_u \cap \mathcal{I}_{\boldsymbol{R}}(u) = \emptyset$. In this thesis, this is referred to as recommendation list for *non-visited items*, and accordingly, if the recommendation list may include items that the user has experienced in the past, it is a recommendation list for *visited items*.

Now it is possible to define a recommender system as a mathematical function. A recommender system can be defined as a scoring function $p_i^u \colon \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$ that accurately estimates future preferences based on information about users' past actions. The score p_i^u represents the appreciation of user u for item i. Therefore, a recommender system can be used to predict the items that are the most likely to be purchased in the future.

An example of implicit users' preference matrix $\boldsymbol{R}$ is illustrated in Table 1, where the rows $u_1, \dots, u_{10}$ represent ten ($M = 10$) different users, and the columns $i_1, \dots, i_5$ represent five ($N = 5$) different items. The ratings represent whether users u have experienced item i ($r_i^u = 1$ if has, and $r_i^u = 0$ if not). Typically, the number of users is large compared to the number of items ($M \gg N$), and the preference matrix $\boldsymbol{R}$ is sparse. By using the notation presented above, the set of users who have experienced item i_3 can be represented as $\mathcal{U}_{\boldsymbol{R}}(i_3) = \{u_2, u_3, u_5\}$, and the set of items experienced by user u_4 is $\mathcal{I}_{\boldsymbol{R}}(u_4) = \{i_1, i_4\}$. No users have experienced item i_5, and thus $\mathcal{U}_{\boldsymbol{R}}(i_5) = \emptyset$. This represents the cold-start problem; how should item i_5 be recommended? In addition, cold-start problem exists for user u_7, who has no experience from any of the items; how should items be recommended to u_7?

Table 1: An illustration of implicit users' preference matrix $\boldsymbol{R}$.

	$\boldsymbol{i_1}$	$\boldsymbol{i_2}$	$\boldsymbol{i_3}$	$\boldsymbol{i_4}$	$\boldsymbol{i_5}$
$\boldsymbol{u_1}$	0	1	0	0	0
$\boldsymbol{u_2}$	0	1	1	1	0
$\boldsymbol{u_3}$	1	1	1	0	0
$\boldsymbol{u_4}$	1	0	0	1	0
$\boldsymbol{u_5}$	0	1	1	0	0
$\boldsymbol{u_6}$	0	1	0	0	0
$\boldsymbol{u_7}$	0	0	0	0	0
$\boldsymbol{u_8}$	0	0	0	1	0
$\boldsymbol{u_9}$	1	1	0	1	0
$\boldsymbol{u_{10}}$	0	1	0	0	0

In general, a recommender system focuses on producing a list of recommendations for a certain user. Recommendations can be made with the help of algorithms. A general framework for an algorithm that produces recommendation list $\mathcal{L}_u$ for a certain user u is shown in Algorithm 1.

Algorithm 1: A general framework for a recommender algorithm

```
Choose the number of candidate items $D$ that will be taken from the full item set $\mathcal{I}$ (positive integer, $D \leq N$).
Choose the size of the recommendation list $L$ (positive integer, $L \leq D$).
1. Choose a subset of items $C \subseteq \mathcal{I}$ by using business-specific criterion. The number of items in $C$ must satisfy $|C| = D$ and $C \cap \mathcal{I}_{\boldsymbol{R}}(u) = \emptyset$.
2. Calculate score $p_i^u$ for each item $i \in C$, where higher score means higher appreciation.
3. Select the top $L$ items based on the highest $p_i^u$. List them in the recommendation list $\mathcal{L}_u$ with respect to some ranking algorithm.
4. Return $\mathcal{L}_u$.
```

2.1.3 Goals

The movement towards providing products and services to customers through electronic commerce has allowed companies to provide customers with more options. The amount of information that customers must process before finding an item that meet their needs has expanded (Schafer, et al., 1999). For users, the goal of a recommender system is often to solve this information overload problem (Resnick & Varian, 1997; Burke, 2007). Recommender systems can also be useful for the users in the following scenarios (Ricci, et al., 2011; Herlocker, et al., 2004):

- **Finding (all) good items**. Recommender systems try to provide a list of items that are of use for the users. Sometimes the list provided may include item combinations, item sequences, or all the possible items that satisfy the needs of a certain user. Recommendation list with predicted ratings or scores can also be helpful for the user.
- **Receiving annotations**. Users can be given personalized annotations in context, such as a TV program recommender system which sends notifications to

users when a TV show worth watching (based on users' preferences) will be aired in the near future.

- **Wanting to just browse**. In this case, users are given the chance of browsing the item catalogues freely, with no intention on purchasing any items. The purpose of the recommender is to browse the items that are more likely to be interesting for the users, provided by the recommender system.
- **Interacting with the recommender system**. Instead of trusting in recommendations coming from recommender systems, some users might just want to test the goodness of the recommendations or the behavior of the system. Users may also be given the possibility to interact with the recommender system and make the recommendations more personalized.
- **Wanting to express themselves**. In some cases, users are more interested in expressing their opinions and beliefs through comments and ratings. In addition, users may want to contribute in order to just help or influence others.

The role that a recommender system plays for a service provider is different from the role it plays for users. Service providers may use recommender systems for numerous reasons, such as (Schafer, et al., 1999; Ricci, et al., 2011; Schafer, et al., 2001):

- **Increasing sales volume**. One of the most common reasons of commercial recommender systems is to be able to sell more with the help of recommendations. This goal is attained by recommending items that the users are most likely to buy.
- **Selling items more diversely**. The service provider might want to sell a variety of items from the catalogue, not only the most popular ones. Diversification can be achieved by recommending items that would be difficult for users to find without recommendations.
- **Understanding the user better**. By gathering explicit and implicit user preference data, the service provider will be able to understand and predict the user behavior. The preference data can be used for various things, such as increasing user satisfaction and loyalty, turning browsers into buyers, or choosing new items for product selection.

The lists above describes how recommender systems can serve multiple purposes. The numerous possibilities for recommender systems have initiated the need for different sources of knowledge and *recommendation techniques* (Ricci, et al., 2011; Gündüz-Ögüdücü, 2010).

2.1.4 Recommendation techniques

Recommendation techniques are used to predict items that might be of use for the users. Several different techniques have been proposed as the basis for recommender systems, such as *collaborative filtering*, *content-based*, *hybrid-based, knowledge-based*, *demographic-based*, and *utility-based* recommendation techniques (Burke, 2007; Bhuiyan, 2013). This is a broad categorization of recommender systems by different techniques, and it is not by any means the only way to categorize recommender systems (Parsons, et al., 2004). The previously mentioned six techniques are briefly summarized in the following.

Collaborative filtering technique builds a database of user preferences for items (Sarwar, et al., 2001). The technique is based on the observation that human interests and preferences are typically correlated, and it assumes that users' past behavior will tend to agree also in the future (Barbieri, et al., 2014). Collaborative filtering exploits this correlation by recommending the target user those items that other users with similar preferences have liked in the past (Schafer, et al., 2001).

Content-based filtering is based on similarities between item contents and features. Content-based recommender systems recommend items that are similar to those that the target user have liked in the past (Pazzani & Billsus, 2007).

Vector Space Models (VSM) can be used for spatial representations of textual item contents, and it allows for representing each item in an n-dimensional space, where each dimension usually corresponds to a term in an overall vocabulary. By representing user profiles and items in the same vector space, recommendations can be derived by calculating similarities. Cosine similarity is the most widely used similarity measure to describe the proximity of two vectors $\boldsymbol{v_1}, \boldsymbol{v_2}$ (Lops, et al., 2011):

$$sim(\boldsymbol{v_1}, \boldsymbol{v_2}) = \frac{\boldsymbol{v_1} \cdot \boldsymbol{v_2}}{\|\boldsymbol{v_1}\| \|\boldsymbol{v_2}\|}. \tag{2.1}$$

In knowledge-based filtering, a recommender system provides a list of recommendations based on the knowledge about users and items. Knowledge-based recommenders use reasoning for finding the products that meet the target user's requirements. These systems often interact with the user to gain the knowledge, and the recommendations are not based solely on user ratings (Bhuiyan, 2013).

In demographic-based filtering**, r**ecommender systems recommend items based on user demographics. These demographics can include a variety of users' personal attributes, such as age, sex, location, education, and occupation. Demographic-based recommender systems try to learn the associations between items and people demographics (Bhuiyan, 2013).

In utility-based filtering, recommendations are based on utility of each item for the target user. The utility is calculated by first using item features as background data, and then determining the utility functions from items to user preferences. Finally, item rankings are calculated with the help of the utility functions (Bhuiyan, 2013).

Hybrid-based recommender systems combine multiple recommendation techniques together, such as all the techniques presented above. In addition to combining different techniques, hybrids can combine various concepts, such as features from different knowledge sources, recommender systems themselves, or different implementations of the same recommendation technique (Burke, 2007).

Graph-based recommender systems can combine both collaborative- and content-based filtering techniques. Graphs present relations between users and items as a bipartite graph. Users and items are usually defined as nodes, and edges connect users and items. Edge weights can be used for representing the strength of users' preference towards the items (Huang, et al., 2002).

2.1.5 Challenges and considerations

Research has shown that recommender systems can help users to make much better decisions with less effort (Häubl & Trifts, 2000). On the other hand, survey findings (ChoiceStream, 2008) have also shown that more than one-half of product recommendation system users are not happy with the recommendations on electronic commerce sites (Yoo, et al., 2013). When building a recommender system, there are many things to consider from the user's perspective, and also from the practical point of view. For example, the type of data available, performance of the system, and the desired scalability and quality of recommendations should be considered (Bobadilla, et al., 2013). The following lists some of the most relevant considerations in the context of this thesis:

- **Sparsity**. Data sparsity arises when users rate a limited number of items. As mentioned in Section 2.1.2, the number of users is typically remarkably larger than the number of items ($M \gg N$), and therefore, the rating matrix $\boldsymbol{R}$ is exceptionally sparse. Sparsity is problematic especially when using collaborative filtering because recommendations are based on aggregating like-minded user preferences (Guo, 2012).
- **Cold start**. As defined in Section 2.1.2, cold start problem arises when almost nothing is known about the user preferences, or when recommendations are required for items that no users have rated. Research on these problems has mainly focused on the latter, where content-based filtering (computing item-item similarities) has been proposed as one possible solution. *Expectation Maximization* (EM) technique has been proposed to solve the user-side cold start problem. Both types of cold-start problems are problematic when using collaborative filtering due to insufficient information (Schein, et al., 2002; Lam, et al., 2008).
- **Item churn**. Some recommender systems are implemented in a dynamic environment, where items are added and removed continuously. Recommender systems need to be able to adapt to these dynamic environments (Barbieri, et al., 2014).
- **Short- and long-term preferences**. Users may have different short- and long-term preferences that a recommender system should take into account. Generally, recommender systems are either focused on building a long-term user profile or making recommendations based on user's short-term preferences (Ricci, et al., 2011).
- **Recommending the same items repeatedly**. Beel et al. (2013) conducted a study to find out how often (if at all) it is reasonable to make same item recommendations to same users multiple times. They found out that generally it makes no sense to display recommendations to the same users multiple times. However, it was also found that users might miss interesting recommendations, if they are shown only once (Beel, et al., 2013).
- **Personalization-privacy trade-off.** Personalization techniques aim to improve the end-user experience by supporting users in filtering, sorting, and classifying information. However, there is a trade-off between user personalization and privacy. In order to receive more personalized recommendations, users need to be willing to sacrifice some level of their privacy (Uchyigit & Ma, 2008).

2.2 Online auctions

Traditional *auctions* are the oldest forms of economic exchange (Samuelson, 2014). There are a variety of different selling institutions that can be defined as an auction. A common aspect of auction-like institutions is that they are anonymous, and they elicit information in the form of bids, for example, who wins what and pays how much is determined on the basis of received information. Auctions are universal, and they may be used to sell any good (Krishna, 2002).

Online auction is an auction, where transactions take place on an Internet portal, and the transactions are negotiated between buyers and sellers (Jank & Shmueli, 2010). Online auctions are typically deadline auctions, where the person with highest standing bid before a fixed stopping time, is declared the winner (Krishna, 2002). The first online auctions were held electronically via email messages, discussion groups, and news groups in 1995. Thus, online auctions are relatively new. Today, there are many well-known online auction sites, such as *eBay*[7] and *uBid*[8], where buyers and sellers can exchange goods and information. In recent years, online auctions have become popular for many reasons: online auction Web sites are constantly available, geographical constraints are negligible, product selections are extensive, and auctions provide entertainment (Jank & Shmueli, 2010).

Moreover, the empirical research of online auctions is thriving, and the research of online auctions has been thriving even more than conventional, brick-and-mortar auctions (Jank & Shmueli, 2010). The research of recommender systems is usually focused on business-to-customer Web sites rather than customer-to-customer which online auction sites are based on (Li, et al., 2007).

The following sections introduce different online auction formats and auction characteristics.

2.2.1 Auction formats

This section presents the most common online auction formats. Online auction sites may provide both single-item auctions where only a single item is up for sale, and multiple item auctions, where multiple items are up for sale simultaneously.

[7] http://www.ebay.com
[8] http://www.ubid.com

The single-item online auction formats presented are the *English auction*, *Reserve auction* and *Fixed Price* auction, and the multiple-item auction formats are *multiple-item Fixed Price auction* and the *multiple-item Dutch auction* (eBay, 2015, p. Overview Of The Different eBay Auction Types; Sanoma, 2015, p. Ohjeet).

English auction is the most traditional auction, where the bidders compete against each other by raising each other's bids until the deadline is reached. The bidder with the highest (and last) standing bid before the deadline is declared as the winner of the auction. The winner pays the seller an amount equal to the price of the last standing bid, and receives the auction object (Samuelson, 2014; Jank & Shmueli, 2010). It is possible that the online auction allows bids also after the deadline, as long as the time since the last bid is under some fixed time limit (for example 5 minutes). The seller may also set the minimum starting bid (Sanoma, 2015).

Reserve auction prevents the seller from selling the item for less than a certain price. This price is called the *reserve price*, and it is set by the seller at the point of putting the item up for auction. The price is only visible to the seller, and the price is not revealed to the bidders until at the end of the auction. If the highest bid is greater than the reserve price at the end of the auction, the bidder pays the seller an amount equal to the price of the last standing bid, and receives the auction item. If the highest bid is smaller than the reserve price, no transactions are made. Despite the seller's reserve price, the reserve auctions work similar to English auctions (eBay, 2015, p. Overview of the Different eBay Auction Types).

In Fixed Price auction (also known as "*Buy Now*" auction), the seller sets a fixed price and a deadline for the auction item. The item is on sale for the fixed price until the deadline is reached. If only a single item is on sale, the first bidder that offers the fixed price before the deadline wins the auction. After that, the bidder pays the seller an amount equal to the fixed price that the seller was asking. If multiple items are on sale, the auction remains open until all the items have been bought for the fixed price. If no bidders are willing to pay the fixed price before the deadline, the auction closes and the item remains unsold. In Fixed Price auctions, there is no need for minimum starting bids (Sanoma, 2015).

Multiple-item Dutch auctions are rare online auctions, where the seller is selling more than one of a certain item with a deadline. In Dutch auction, the buyers bid a price and say how many items they are willing to buy. When the deadline is reached, everyone pays the lowest price that was bid by one of the winning bidders and receives the item or items (eBay, 2015, p. Overview of the Different eBay Auction Types).

Sellers may also be given the possibility to combine Fixed Price with other auction formats, such as Fixed Price and Reserve auction (Fixed Price Reserve auction), or Fixed Price and English auction (Fixed Price English auction). Fixed Price English auction works similar to normal English auction, but it also allows bidders to buy the auction item instantly by offering the fixed price. If no bidder offers the fixed price before the deadline, the auction ends up being a normal English auction, and if the fixed price is offered, the auction closes as it normally would in a Fixed Price auction. The Fixed Price Reserve auction works likewise (Sanoma, 2015).

2.2.2 Special characteristics of online auctions

A recommender system in an online auction environment facilitates trading by helping the buyers to find suitable items from the sellers. In this case, the role of a recommender system can be to build and retain user relationships, and promote sales (Li, et al., 2007). In order to know the type of items that should be recommended to different online auction users, it is important to understand auction characteristics and user preferences that guide users in their decision-making.

The research (Drake, 2007) on online auction characteristics separates *auction selection* and *product valuation* as two separate decision-making processes of the user. The first describes how users choose between different auctions (thus, items), and the latter describes the decision of determining how much to bid on a certain item. Table 2 presents the five most and least important auction characteristics for both auction selection and product valuation, according to the study conducted by Drake (2007). In addition to Table 2, some other characteristics that were included in the study were shipping costs, shipping options, return policy, payment methods accepted, reserve price, minimum bid, “*Buy now*” option, and seller feedback.

Table 2: The most and least important auction characteristics.

#	**Most important** (auction selection)	**Most important** (product valuation)	**Least important** (both)
1	Photo of the item	Photo of the item	Seller location
2	Item quality	Item quality	Proxy bidding[9]
3	Item description	Item description	Shipping insurance
4	Security	Current bid	Rate of bidding
5	Time remaining	Time remaining	Number of bidders

Another study by Drake et al. (2015) used signal theory methods to explore different factors that guide buyers in their online auction decision making (Drake, et al., 2015). These are presented in Table 3.

Table 3: Factors explained by auction characteristics.

Factor loadings (variance explained)	**Factor**	**Auction characteristics**
25 %	Item quality	Item quality Item description Photo of product
12 %	Logistics	Shipping costs Shipping options Shipping insurance Payment methods accepted
8 %	Competition	Time remaining Rate of bidding Number of bidders
5 %	Minimum price	Reserve price Minimum bid
4 %	Service expectations	Return policy Seller location
4 %	Expected winning bid	Current bid Time remaining
3 %	Reputation	Feedback scores
3 %	Default purchase	“Buy now” option

[9] Proxy bidding: When the highest bidder bids, the winning price is always a small increment (often determined by the seller) above the next lowest big (Roth & Ockenfels, 2002).

Online auction transactions are often paid-up-front, which makes the buyers vulnerable to frauds. Therefore, online auctions use feedback systems to give users a possibility to build their reputation by giving each other feedback. Because negative feedbacks are very rarely given, its consequences on users' overall reputation are more significant than positives. Therefore, users may try to avoid the consequences of negative feedback by creating multiple accounts and pseudonyms (Wang & Chiu, 2008; Resnick, et al., 2006).

There is evidence from eBay and Amazon online auctions that the decision dynamics change near the end of an auction. This is called *last-minute bidding*, also known as *sniping*, where the bidders submit their bids near the closing seconds of the auction. There is not just one reason for sniping, but it is the best response in many bidding strategies. For example, bidding early could give a signal to other buyers that the item is exceptionally valuable, and thus raise the price. Online auctions can deal with sniping by giving every bidder a chance to bid, even after the deadline, as long as the time since the last bid is under some fixed time limit (Roth & Ockenfels, 2002).

Online auctions also deal with different types of sellers. The sellers may have large diversity in their company types, product assortments, objectives and strategies. Especially the corporate sellers may have different strategies, such as finding new customers, increasing profit margins, or gaining reputation. Therefore, auction sites often charge a listing price from the corporate sellers (Becherer & Halstead, 2004).

In addition to previously mentioned characteristics, online auctions are very dynamic marketplaces, where the items are unique and short-lived. Moreover, there are usually many bidders willing to buy certain item, whereas only one of them can win the item. Often those who made a bid but did not win the item are willing to buy the same item or a similar item from someone else. On the other hand, the one who won the item, is not likely wanting to buy a similar item again (Katukuri, et al., 2013; Pinckney, 2013a).

2.3 Recommender systems in practice

The purpose of this section is to overview recommenders that are in real-world use on popular Web sites, such as YouTube and eBay. It should be noted that the information presented in this section may not be up-to-date, and due to the speed of change, the recommenders currently implemented on those sites may differ from what is presented here. Moreover, detailed information on the techniques and algorithms in use are challenging to find, which may be due to the business value of the recommenders.

2.3.1 YouTube

YouTube is a popular online video community, where the recommender recommends video sets to its users. The goal of the recommender is to provide very high quality and personalized video recommendations based on both content and users' activity data. Content data include raw video streams and video metadata, and users' activity data can be further divided into explicit and implicit data. Explicit data include users' ratings, favorites, likes and subscriptions, whereas implicit data is generated from activities like viewing times and interaction with videos (such as the proportion of video that a user watched). The following is based on YouTube's recommendation system that was in use in 2010 according to Davidson et al. (2010).

One of the main techniques that YouTube use is association rule mining, where the number of co-visitation counts c_{ij} are calculated for each pair of videos (v_i, v_j). Then, a score called *relatedness* r_{ij} is calculated with the help of the following formula

$$r(v_i, v_j) = \frac{c_{ij}}{f(v_i, v_j)}, \tag{2.2}$$

where $f(v_i, v_j)$ is a normalization function, such as $f(v_i, v_j) = c_i \cdot c_j$. Now, given a seed video v_i, videos $v_j, j \neq i$ can be ranked in a decreasing order with respect to the relatedness measure, and top N candidate videos can be chosen in ranking R_i. These related videos in R_i can be considered as a directed graph: for each pair of videos (v_i, v_j), there is an edge e_{ij} that connects v_i and v_j only if $v_j \in R_i$, where the weight of the edge is given by (2.2).

After this, a candidate video set will be generated. The generation begins by generating a set of videos based on user's personal activity on the site. This is called a *seed set* S, which contains videos that the user has shown activity towards (such as favorited, rated, or added to playlist). The relatedness rankings R_i are calculated for each of the videos in the seed set, $v_i \in S$, and the union of these rankings is denoted as

$$C_1(S) = \bigcup_{v_i \in S} R_i. \tag{2.3}$$

Taking only C_1 as a set of candidate videos often leads to a narrow set of related videos, which is why this candidate set is expanded by taking a limited transitive closure over the related videos graph. Define C_n as the set of videos that are reachable in n steps from any video in the seed set

$$C_n(S) = \bigcup_{v_i \in C_{n-1}} R_i, \tag{2.4}$$

where C_{n-1} is a recursive definition, and $C_0 = S$ is the first set. By removing the original seed set S from this, and taking all the cases C_i from $i = 0$ to N_c, the candidate set for recommendations is obtained as

$$C_f = \left(\bigcup_{i=0}^{N_c} C_i \right) \setminus S. \tag{2.5}$$

Because the size of the candidate set is large, the next step is to rank the videos in the set. The ranking is a three-step procedure, and it is done by using a linear combination of the following *signals*: 1) video quality (assure video quality by using view count, ratings, and such), 2) user specificity (boost user's unique taste and preferences), and 3) diversification (remove too similar videos from the ultimate list). Based on the computed rankings, a small number of recommendations (from 4 to 60) are displayed through the user interface (Davidson, et al., 2010).

In the previous it was described how a number of techniques from Section 2.1.4 are used in the YouTube recommender. Therefore, this recommender could be classified as a hybrid recommender that uses for instance collaborative-, demographic-, content-, and graph-based techniques. The YouTube recommender also follows the general framework of a recommender algorithm (see Algorithm 1), where a set of candidate items are first chosen, then ranked, and finally displayed.

2.3.2 eBay

eBay is a multinational online auction site where buyers and sellers can exchange items and information. There is little information about the current recommender system that eBay is using, but Tom Pinckney held a speech about eBay's graph-based recommender system in 2013. This system is explained in the following, based on the video (Pinckney, 2013a) and slides (Pinckney, 2013b) of Pinckney's speech.

The recommender system at eBay is based on user's personal *taste profile*, which describes the set of things that the user likes, and the set of things that the user does not like. Taste profile is based on the assumption that likes and dislikes are correlated. The purpose of using taste profiles is to reduce the computation time by reducing the dimension of the problem. Back in 2013, eBay had graphs with 40 billion edges, 2 billion item nodes, and 200 million user nodes.

The graph technique is explained with the help of the following example. First, assume user A, who likes bicycles and carrots (items). On the other hand, assume user B who dislikes bicycles and carrots but likes cars. Now, based on the taste profile's correlation assumption, it is possible to infer that another user C, who dislikes carrots, would like cars over bicycles. This is due to the fact that user B also dislikes carrots, and therefore user C would choose similar to B. The situation is illustrated in Figure 2.

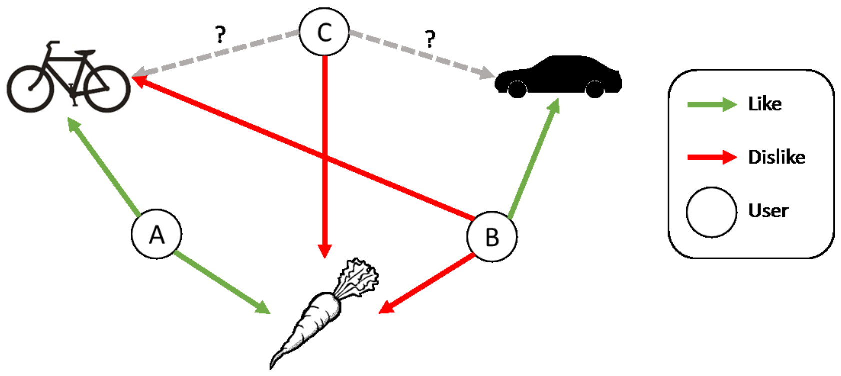


Figure 2: Inferring correlations in eBay's recommender system. User C would like the car instead of the bicycle because both users B and C dislike carrots.

In the previous scenario, information about user B and C's preferences on certain items were used in order to predict user C's choice. Assuming more users (D, E, F, and G), and that they can describe their preferences from -2 (dislike) to 2 (like) on each item, it is possible to plot the users' preferences in Figure 3. It can be seen that the users almost form a straight line, and there seems to be a hidden factor, a *latent factor,* that could explain users' preferences in less dimensions. For example, one reason why users are either liking or disliking both carrots and bicycles could be that some users are more ecological than others. Therefore, it could be possible to describe user preferences to some extent by using only the ecological factor rather than all the items themselves (Ecological axis in Figure 3). In general, latent factors can be inferred but not observed, and there is not necessarily a reasonable real-world interpretation for the factor. The exact techniques that eBay uses for extracting the latent factors were not introduced in the source material.

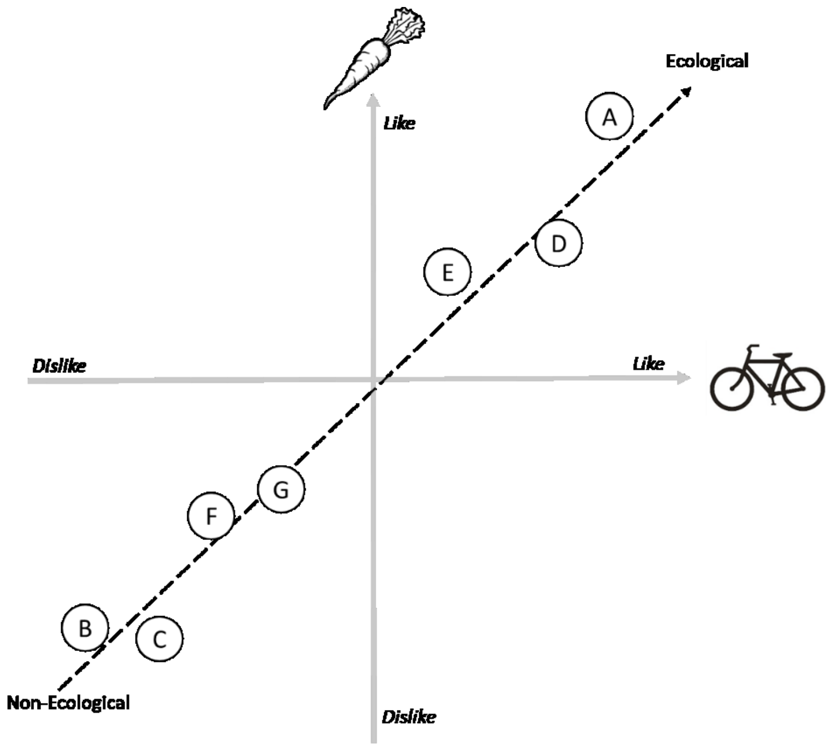
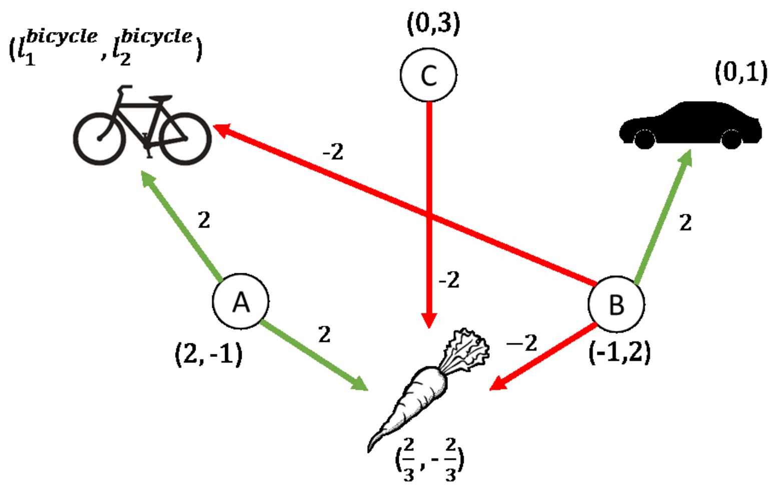


Figure 3: Users' preferences plotted in two dimensions, illustrating how much users like bicycles and carrots.

In general, eBay uses a number of latent factors l_i in order to create an n-dimensional *taste space* $T = (l_1, \ldots, l_n)$. The taste space is used for creating a taste profile $T^u = (l_1^u, \ldots, l_n^u)$ for each user u. Therefore, the taste profile consists of latent factor coordinates $l_i^u, i = 1, \ldots, n,$ also known as *taste coordinates*. Moreover, items i are described in terms of the taste coordinates, $T^i = (l_1^i, \ldots, l_n^i)$ (such as how ecological is the carrot compared to the bicycle).

Now, assume a graph as in Figure 2, where items and users are the nodes, and edges can only connect users and items. Moreover, assume only two latent factors l_1 and l_2. By defining edge weights from users to an item to be the cross product of taste profiles between users and the item, it is possible to constrain similar items to be close to each other. The situation is presented in Figure 4, where the taste profile of the bicycle can be calculated by solving

$$\begin{cases} -1l_1^{bicycle} + 2l_2^{bicycle} = -2 \\ \ \ 2l_1^{bicycle} - 1l_2^{bicycle} = 2 \end{cases} \rightarrow T^{bicycle} = \left(\frac{2}{3}, -\frac{2}{3}\right). \qquad (2.6)$$



$(l_1^{bicycle}, l_2^{bicycle})$
(0,3)
C
(0,1)
-2
2
2
-2
A
B
(2,-1)
2
−2
(-1,2)
$(\frac{2}{3}, -\frac{2}{3})$

Figure 4: Calculating the taste profile $T^{bicycle} = (l_1^{bicycle}, l_2^{bicycle})$ for the bicycle.

The users' taste profiles are updated as users like (for example purchases items) or dislike (for example views the item's Web site and ignores the item) items. Taste profile coordinates are updated so that only the coordinates of the item and user in question are updated. This requires checking that every adjacent nodes cross products will remain the same as before. For example, when user C in Figure 4 buys (likes) the car, only user C's and the car's coordinates are updated. In this example, there are many possible coordinate combinations, and the solution is not unique. One possible solution is presented in Figure 5.

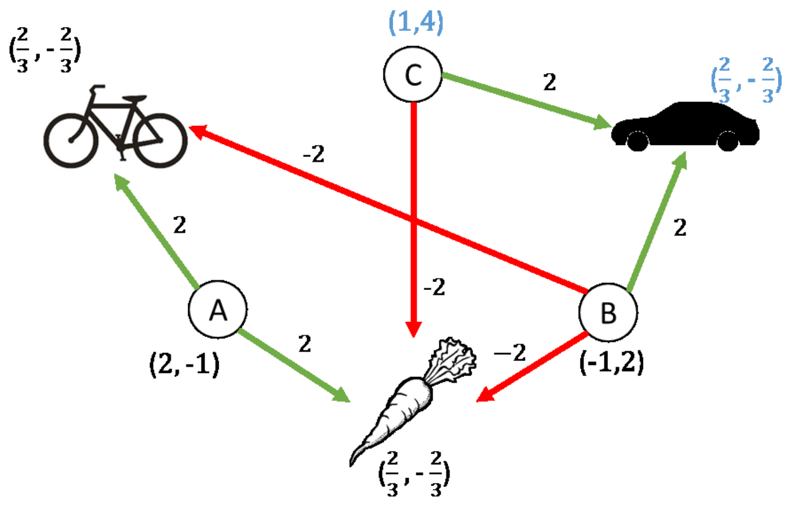

$(\frac{2}{3}, -\frac{2}{3})$
(1,4)
C
2
$(\frac{2}{3}, -\frac{2}{3})$
-2
2
2
-2
A
B
2
−2
(2,-1)
(-1,2)
$(\frac{2}{3}, -\frac{2}{3})$

Figure 5: Updating the taste profile coordinates. Only the coordinates of the item and user in question are changed (marked with blue color).

As mentioned in Section 2.2.2, online auctions are highly dynamic marketplaces, where items are added and removed constantly. eBay tackles this with the help of graphs and low-dimensional users' taste profiles, rather than computing large and dynamic item-item or user-user similarity matrices. Recommendations for a certain user can be computed by calculating the distances between user's taste profile and item profiles. If necessary, too similar items are not recommended.

The exact techniques how eBay generate the end-user recommendation list from the graphs are not revealed in the speech. It remains unknown how the latent factors or possible candidate items are generated, if at all. However, the key idea in their recommender is to represent both users and items in the same vector space, and calculate similarities in this space to generate the recommendations. Dimensionality reduction is used to reduce the computation time.

3 Evaluating recommender systems

Numerous goals for recommender systems were introduced in Section 2.1.3. It is useful to evaluate how well recommender systems perform with respect to these goals, and also with respect to variety of properties such as accuracy, robustness and scalability (Shani & Gunawardana, 2011). This chapter describes experimental settings and evaluation metrics that help in making choices between different recommender systems.

3.1 Experimental settings

Recommender systems can be compared with the help of experimental methodology. The methodology follow guidelines that are common for empirical studies: a hypothesis must be set before running the experiment, all sources of variation must be identified, and the generalization power of the experiment must be considered when drawing conclusions (Dean & Voss, 1999; Shani & Gunawardana, 2011). Shani and Gunawardana (2011) divide experimental settings into *user studies*, *offline experiments* and *online experiments* (Shani & Gunawardana, 2011). The latter two are the most relevant in the context of this thesis, and they are discussed in the following.

Offline evaluation is usually based on simulating the online environment of a recommender system by splitting the available data into training and test sets, similar to machine learning algorithms (Barbieri, et al., 2014; Bishop, 2006). In recommender evaluation, the original dataset can be split into training and test sets by splitting in time, and using these sets the recommender can be simulated by making predictions for the unseen test set. However, in order to make valid deductions from the simulations, it needs to be assumed that the user behavior would have been the same whether the recommender was actually in use or not (Shani & Gunawardana, 2011). Offline experiments are attractive since they require no interaction with the users. However, the drawback is that offline evaluations can only provide metrics from the past, and therefore, only a very few questions can be answered by solely relying on offline experiments (Shani & Gunawardana, 2011; Barbieri, et al., 2014).

In an online evaluation, recommender systems are partly or fully deployed in the target environment, where the recommender systems can be used by real users. This allows for testing the effects of the recommender system on real users that interact with the system. In addition, online evaluations are useful for testing multiple different recommender systems simultaneously to compare and rank them. Online evaluations require careful planning in order to draw reliable conclusions. For example, users need to be sampled multiple times and randomly when dividing them into subsets of users in order to compare different recommender systems. Online evaluations can also be expensive and risky, which is why it is reasonable to develop algorithms first by using offline evaluation (Shani & Gunawardana, 2011).

3.2 Evaluation properties

Shani and Gunawardana (2011) have listed numerous recommender system properties for measuring and ranking different recommender approaches (Shani & Gunawardana, 2011). Unless mentioned otherwise, the following is based on this list with the help of notation from Section 2.1.2.

3.2.1 Predictive accuracy

Predictive accuracy measures the accuracy of predicted ratings produced by the recommendation system. The accuracy can be measured by comparing the predicted ratings $\hat{r}_i^u$ and actual ratings r_i^u with respect to some metric. Predictive accuracy is usually calculated in offline experiments, since the real ratings r_i^u are typically known. In offline experiments, it is possible to choose a test set $\mathcal{T}$ that contains only user-item pairs $\langle u, i \rangle$ with an actual rating $r_i^u \in \boldsymbol{R}$ (thus, $\mathcal{I}_{\boldsymbol{R}}(u) \neq \emptyset$ and $\mathcal{U}_{\boldsymbol{R}}(i) \neq \emptyset$ for all $\langle u, i \rangle \in \mathcal{T}$), and then calculate the accuracy with respect to some metric. Predictive accuracy can be calculated in a similar manner when performing online experiments, where the only difference is that the real ratings are usually retrieved through users' online usage.

A typical framework for an algorithm that measures predictive accuracy is presented in Algorithm 2.

Algorithm 2: A typical framework for measuring predictive accuracy

1. Divide user-item pairs $\langle u, i\rangle \in \boldsymbol{R}$ into a training set $\mathcal{S} \subset \langle u, i\rangle \in \boldsymbol{R}$ and a test set $\mathcal{T} \subset \langle u, i\rangle \in \boldsymbol{R}$, so that $\mathcal{S} \cap \mathcal{T} = \emptyset$, $\mathcal{S} = \mathcal{T} \neq \emptyset$ and $\mathcal{T} \cap \langle \mathcal{U}_{\boldsymbol{R}}(i), \mathcal{I}_{\boldsymbol{R}}(u)\rangle \neq \emptyset$ for all $\langle u, i\rangle \in \mathcal{T}$. This means that the test set contains only user-item pairs that have an actual rating, and the training set consists of the rest of the user-item pairs.
2. Train the recommender system with training set $\mathcal{S}$.
3. Predict ratings $\hat{r}_i^u$ for all user-item pairs $\langle u, i\rangle \in \mathcal{T}$. Because a recommender system was defined (Section 2.1.2) to be a scoring function $p_i^u\colon \mathcal{U} \times \mathcal{I} \to \mathbb{R}$ that maps user-item pairs to users' preferences, the output of the function can be interpreted as users' predicted ratings $\hat{r}_i^u$.
4. Compare predicted ratings $\hat{r}_i^u$ and the actual original ratings r_i^u for all $\langle u, i\rangle \in \mathcal{T}$ with respect to some metric.

The *Root Mean Squared Error* ($RMSE$) is a metric that measures how close the actual and predicted ratings are to each other with the following formula:

$$RMSE = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{\langle u,i\rangle \in \mathcal{T}} (\hat{r}_i^u - r_i^u)^2}, \tag{3.1}$$

where $|\mathcal{T}|$ is the number of user-item pairs $\langle u, i\rangle$ in the test set $\mathcal{T}$.

The *Mean Absolute Error* (MAE) measures the absolute deviation between the actual and predicted rating:

$$MAE = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{\langle u,i\rangle \in \mathcal{T}} |\hat{r}_i^u - r_i^u|}. \tag{3.2}$$

Instead of trying to predict the exact ratings, one could try predicting the items that a user would choose. This is called *usage prediction*, where the idea is to hide some of the items $i \in \mathcal{I}_{\boldsymbol{R}}(u)$ that a user u has experienced in the past, and then observe whether the recommender system recommends these items for the user. The logic for this is presented in Algorithm 3.

Algorithm 3: A typical framework for measuring usage prediction

Assumptions: Let $\boldsymbol{R}$ be a binary matrix of active users' $u \in \mathcal{U}$ preferences, where each element $r_i^u \in \boldsymbol{R} \in \{0, 1\}$ represents whether a user has experienced an item $i \in \mathcal{I}$. If user u has experienced item i, then $r_i^u = 1$, and if not, then $r_i^u = 0$. Let $\mathcal{I}_{\boldsymbol{R}}(u)$ be the set of items that a user u has experienced, and $\mathcal{U}_{\boldsymbol{R}}(i)$ the set of users that have rated item i.

Initialize list $\mathcal{L} = \{\}$, which will be a list of lists that contain recommended items for each user u.

Choose N_r to be number of recommendations that will be suggested for each user u.

```
1. Select user-item pairs $\langle u, i\rangle \in \boldsymbol{R}$ into a test set $\mathcal{T} \subset \langle u, i\rangle \in \boldsymbol{R}$, so that $\mathcal{T} \neq \emptyset$, $\mathcal{T} \cap \langle \mathcal{U}_{\boldsymbol{R}}(i), \mathcal{I}_{\boldsymbol{R}}(u)\rangle = \emptyset$ for some $\langle u, i\rangle \in \boldsymbol{R}$, and $r_i^u = 1$ for all $\langle u, i\rangle \in \mathcal{T}$. This means that the test set contains only user-item pairs for which a user has experienced the item, but not all of them.
2. Choose a training matrix $\boldsymbol{S} = \boldsymbol{R}$, and modify it so that $r_i^u = 0 \in \boldsymbol{S}$ for all $\langle u, i\rangle \in \mathcal{T}$. This means that the training matrix equals the original rating matrix with some of the items hidden.
3. Train the recommender system with the training matrix $\boldsymbol{S}$.
4. For each user $u$ do:
    4.1. Use the recommender system to produce a list of $N_r$ recommendations $\mathcal{L}_u$ for user $u$ (as in Algorithm 1).
    4.2. Expand list $\mathcal{L}$ with $\mathcal{L}_u$.
    4.3. End for-loop
5. Compare recommendation list $\mathcal{L}$ with the hidden user-item pairs in test set $\mathcal{T}$. Do the comparison with respect to some metric.
```

In offline experiments, usage prediction is based on the assumption that users would make their decisions in the same way, whether the recommender system was actually deployed or not. In other words, it needs to be assumed that the recommender system would not have had an effect on users' choices in the past. This assumption can be false, for example in a situation where an unaware user would have wanted to experience the item, but no chance to see the item was given. Therefore, the recommenders may seem to perform worse in comparison to if they were actually implemented at the time of evaluation.

The comparison of users' experiences on items with the recommendations provided by the recommender system (Algorithm 3, step 5) can be done for each user alone, or to multiple users simultaneously. Table 4 presents four different scenarios that are possible when comparing whether a certain user has experienced a certain item with the items on the recommendation list. If the user has experienced the item and the recommender system recommends this item, it is *True-Positive* (TP), and if the recommender system does not recommend the item, it is *False-Negative* (FN). If the user has not experienced the item and the recommender system recommends this item, it is *False-Positive* (FP), and if the recommender system does not recommend it, it is *True-Negative* (TN).

Table 4: Usage prediction scenarios when comparing whether a certain user has experienced a certain item with the items on the recommendation list.

	Recommended	**Not recommended**
Experienced	True-Positive (TP)	False-Negative (FN)
Not experienced	False-Positive (FP)	True-Negative (TN)

By classifying each item in the user's recommendation list to one of the cells in Table 4, it is possible to count user-specific usage prediction measures. Some of the most relevant measures in this thesis are presented in the following.

The first usage prediction measure is *precision*, which is defined as follows:

$$Precision_{N_r} = \frac{\#TP}{\#TP + \#FP}, \tag{3.3}$$

where $\#TP$ is the number of items that were classified in Table 4 cell TP, and $\#FP$ is the number of items that were classified in cell FP. In a similar manner, $\#FN$ is the number of items classified in cell FN, and $\#TN$ is the number of items classified in cell TN. Because the number of items that are classified in each cell depends on the number of recommendations N_r (see Algorithm 3), precision measure can be calculated for different recommendation list sizes, which is noted by the sub-index $Precision_{N_r}$. Precision describes the proportion of items that were suitable for the user from all the items that were recommended.

Another measure for usage prediction is *recall*:

$$Recall_{N_r} = TPR_{N_r} = \frac{\#TP}{\#TP + \#FN}, \tag{3.4}$$

which is also known as the *True-Positive-Rate* (TPR). The variables in the formula were defined above where the variables of equation (3.3) were defined. Recall is also dependent on N_r, and it is denoted with similar sub-index to precision. Recall describes the proportion of items that were correctly recommended from all the items that would have been suitable recommendations for the user.

There is typically a tradeoff between precision and recall, and longer recommendation lists typically improve recall and reduce precision. *F1-Score* (Rijsbergen, 1979) is a measure that summarizes this tradeoff by taking the harmonic mean of precision and recall:

$$F1 - Score_{N_r} = 2\frac{Precision_{N_r} * Recall_{N_r}}{Precision_{N_r} + Recall_{N_r}}. \tag{3.5}$$

The idea of *ranking measures* is to measure the ordering of items in a recommendation list $\mathcal{L}_u$ that is displayed to a user through a user interface. The ordering of the items is produced by a ranking algorithm of the recommendation system, which aims to order the set of items according to user's preferences (see Algorithm 1, part 3). In a simple case, the ranking algorithm just orders the items in a decreasing order with respect to the predicted ratings.

One possibility for scoring rankings is to use a *reference ranking*, in which the ranking scores are based on correlation of some "true" ranking and the ordering of the recommendation list. When it is only known which items users have visited and which not, the visited items should be ranked above the non-visited items. This is valid only if it is known that the user was aware of all the non-visited items, and the user actually preferred visited items to the non-visited items. Constructing the true ranking is challenging when the number of items is large, and users may only see a fraction of these items.

Another possibility is to use *utility-based* ranking measures, where it is assumed that the utility of a recommendation list is additive, and can be given as a sum of the utilities of individual recommendations. In this case, recommended items are discounted by a factor that depends on its position in the list of recommendations. Usually, it is assumed that users view recommendation lists from the beginning to the end, which is why the utility is discounted more heavily towards the end of the list. This assumption is reasonable when it is expected that users will only view a few of the items at the top of the recommendation list.

R-Score (Breese, et al., 1998) metric assumes that the value of recommendations declines exponentially down the recommendation list, yielding the following score for each user u:

$$R_u = \sum_u \sum_j \frac{\max(r_{ui_j} - d, 0)}{2^{\frac{j-1}{\alpha-1}}}, \tag{3.6}$$

where i_j is the item in the j:th position, r_{ui} is user u's rating for item i, d is a task-dependent neutral rating, and α is a half-life parameter that controls the exponential decline of the position values in the list. The half-life parameter can be interpreted as the number of items in the list such that there is a 50-50 chance the user will review that item. For usage prediction, typically $d = 0$ and $r_{ui} = 1$ if the user has experienced the item, and 0 otherwise. The user-specific scores can be aggregated using

$$R = 100 \frac{\sum_u R_u}{\sum_u R_u^*}, \tag{3.7}$$

where R_u^* is the score of the best possible ranking for user u.

If it expected that user might view a large proportion of the list, a slower decay is needed. In this case, *Normalized Cumulative Discounted Gain* (NDCG) (Järvelin & Kekäläinen, 2002) with a logarithmic discount can be used. Assuming gain g_{ui} for each user u being recommended an item i, the average Discounted Cumulative Gain (DCG) for a recommendation list with N_r items can be defined as

$$DCG = \frac{1}{M} \sum_{u=1}^{M} \sum_{j=1}^{N_r} \frac{g_{ui_j}}{\max(1, \log_{\mathrm{b}} j)}, \tag{3.8}$$

where the logarithm base b is a free parameter. NDCG is the normalized version of DCG

$$NDCG = 100 \frac{DCG}{DCG^*}, \tag{3.9}$$

where DCG^* is the best possible DCG.

3.2.2 Item coverage

Item coverage refers to the extent that items are covered when providing recommendations. Item space coverage can be measured by simply examining the percentage of items that can be recommended by the recommender system, denoted with $\mathcal{I}_P \subseteq \mathcal{I}$. This measure is defined as *Simple Item Space Coverage* ($SISC$):

$$SISC = \frac{|\mathcal{I}_P|}{N}, \tag{3.10}$$

where $|\mathcal{I}_P|$ is the number of items in set $\mathcal{I}_P$, and N is the number of items in set $\mathcal{I}$.

3.2.3 Other properties

Confidence measures how much the system trusts in its recommendations. One of the most common measures for this is the confidence interval, which statistically guarantees that the predicted value from the recommender system lies between a confidence interval with pre-defined probability α. Users can benefit from confidence values when choosing items. For example, users might want to find more information about an item with low level of confidence (thus, the confidence interval is large). Confidence can also be used for pre-filtering low confidence level items before making recommendations to users.

In contrary to confidence, *trust* measures how much a user trusts in the recommendations that are produced by the recommender system. Trust can be measured by conducting user studies, or by assuming that users will use the system repeatedly if they trust the recommendations. Trust requires user interaction, and it cannot be measured in offline experiments.

Some users expect recommender systems to recommend products that they did not know about. *Novelty* of the recommendations can be measured online by conducting user studies, but also offline. In offline experiment, the data is first split by specific point on time, meaning that the user ratings made after that point are hidden. After that, by also hiding some items before that time point, it is possible to simulate that the user was acquainted with these items but did not rate them. In this case, the recommender system is rewarded for recommending items that user rated after the split, and penalized for recommending items that user rated before the split. Moreover, it is possible to measure novelty by using the assumption that popular items are less likely to be novel.

In a scenario where an item was successfully recommended to a user (for example, user decided to click the item that was recommended), it is possible measure how surprising the recommended item was to the user. This measure is known as *serendipity*, which is as a measure for the amount of relevant information that the user was able obtain from the recommendation. In online experiments, serendipity can be measured through user studies, and in offline experiments it can be measured by computing the distances between new successful recommendations and user's previous ratings.

Diversity is described as the opposite of similarity. If the recommender system suggested only similar items to a user, it would be time-consuming for the user to browse through the whole range of items. One of the most explored methods for measuring diversity is by computing item-item similarities with the help of item features (see content-based filtering in Section 2.1.4). The actual measure can be based on various quantities, such as on the minimum, maximum or average distances between item pairs.

Utility measures the benefits of a recommender system for either the system provider or the user. For the system provider, utility can mean various things, such as the generated revenue or increase in average visit time of a user (see goals of a recommender system in Section 2.1.3). It is usually simple to evaluate by computing and comparing the quantity in question between different recommender systems through online experiments. From the user's perspective, utility describes the benefits that a user can gain from the recommendations, which is difficult to model in reality.

The users of a recommender system have their own *risk* profiles. They can be risk-averse or risk-seeking towards recommendations, and for example, some online auction users might be willing to buy items from other users that have had negative feedback in the past, while some not.

Robustness measures the recommender system's ability to tolerate fake information. For example, if a recommender system uses viewing time of an item as an implicit rating, and in reality the user was away from the computer for the whole time, the recommender should not make too strong assumptions of the user's preferences. Moreover, a user could try to boost an item's popularity by using fake profiles, and on the other hand, a user could try to inject competitors' by giving negative ratings. Robustness can be evaluated by experimenting how sensitive the system is to these kind of *attacks* with respective to the system's goals. Another type of robustness is more related to the technical side of the recommendation system, such as how to handle a large number of queries or malfunctions.

Privacy emerges when a recommender system discovers user's preferences, and with the help of these preferences, makes recommendations to other users. This generates a privacy risk, and because collaborative recommendation techniques (see Section 2.1.4) are based on this, the recommender system must be certain not to leak any sensitive private information. In addition, user anonymity is a privacy concern, as seen in the Netflix Price (Amatriain & Basilico, 2012) case, where researchers were able to reveal the anonymity of some Netflix users by combining the anonymized movie rating data from multiple sources (Narayanan & Shmatikov, 2008). Privacy usually comes at the expense of predictive accuracy, and it can be evaluated, for example by comparing different systems with respect to the portion of users whose private information were at risk.

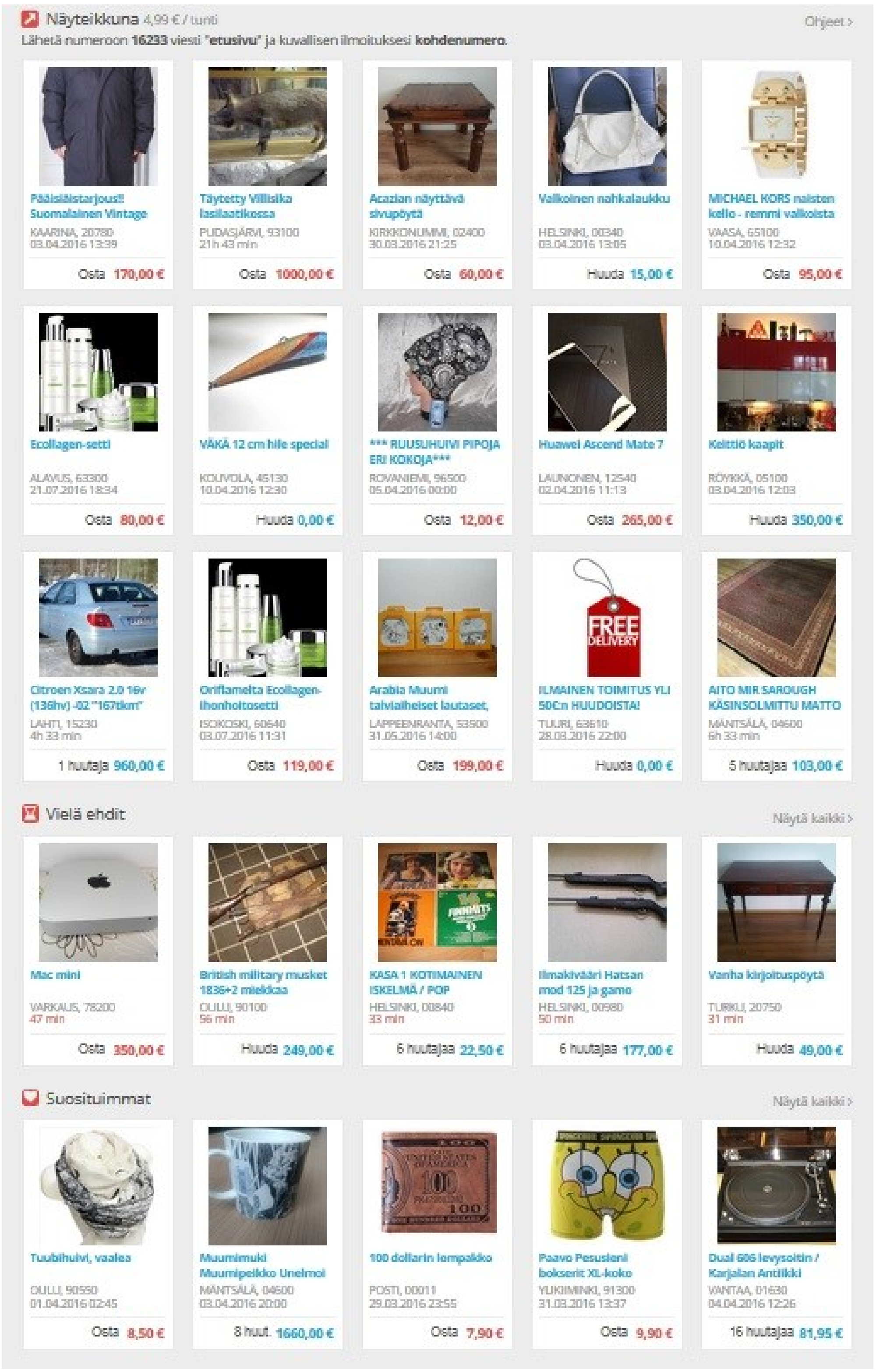
Recommender systems are often implemented at large scale, which is why *scalability* of the system plays an important role. The chosen algorithms and techniques affect the resources that are needed, such as the computational power, memory and time. Users expect recommender systems to provide recommendations rapidly, and therefore the algorithms and techniques must be chosen with respect to available resources. Thus, it is useful to understand the consumption of these resources over large data sets. Scalability can be measured by different quantities, such as number of recommendations per second, or latency (time for making a recommendation online).

4 Developing the recommender

4.1 Problem description

In this thesis, a recommender system is built for a major Finnish online auction site. The recommender system is built and evaluated offline using historical online browsing data of the auction site users.

Figure 6 illustrates how the items are currently displayed on the front page of the online auction site. In general, the items on the front page can be divided into three different main categories: *'Display window'* ('*Näyteikkuna*' in the figure), *'Ending soon'* ('*Vielä ehdit*' in the figure), and '*Most popular*' ('*Suosituimmat*' in the figure). There are 15 items under the category 'Display window', where the sellers have paid money to get their auction item on front page display for a certain amount of time. The category 'Ending soon' aims to attract the last-minute bidders by displaying five auction items that are closing soon, and 'Most popular' displays five of the most popular items based on item view count.



Näyteikkuna 4,99 € / tunti
Ohjeet >
Lähetä numeroon **16233** viesti **"etusivu"** ja kuvallisen ilmoituksesi **kohdenumero**.

Pääsiäistarjous!! Suomalainen Vintage
KAARINA, 20780
03.04.2016 13:39
Osta 170,00 €

Täytetty Villisika lasilaatikossa
PUDASJÄRVI, 93100
21h 43 min
Osta 1000,00 €

Acazian näyttävä sivupöytä
KIRKKONUMMI, 02400
30.03.2016 21:25
Osta 60,00 €

Valkoinen nahkalaukku
HELSINKI, 00340
03.04.2016 13:05
Huuda 15,00 €

MICHAEL KORS naisten kello - remmi valkoista
VAASA, 65100
10.04.2016 12:32
Osta 95,00 €

Ecollagen-setti
ALAVUS, 63300
21.07.2016 18:34
Osta 80,00 €

VÄKÄ 12 cm hile special
KOUVOLA, 45130
10.04.2016 12:30
Huuda 0,00 €

*** RUUSUHUIVI PIPOJA ERI KOKOJA***
ROVANIEMI, 96500
05.04.2016 00:00
Osta 12,00 €

Huawei Ascend Mate 7
LAUNONEN, 12540
02.04.2016 11:13
Osta 265,00 €

Keittiö kaapit
RÖYKKÄ, 05100
03.04.2016 12:03
Huuda 350,00 €

Citroen Xsara 2.0 16v (136hv) -02 ""167tkm"
LAHTI, 15230
4h 33 min
1 huutaja 960,00 €

Oriflamelta Ecollagen-ihonhoitosetti
ISOKOSKI, 60640
03.07.2016 11:31
Osta 119,00 €

Arabia Muumi talviaiheiset lautaset,
LAPPEENRANTA, 53500
31.05.2016 14:00
Osta 199,00 €

ILMAINEN TOIMITUS YLI 50€:n HUUDOISTA!
TUURI, 63610
28.03.2016 22:00
Huuda 0,00 €

AITO MIR SAROUGH KÄSINSOLMITTU MATTO
MÄNTSÄLÄ, 04600
6h 33 min
5 huutajaa 103,00 €

Vielä ehdit
Näytä kaikki >

Mac mini
VARKAUS, 78200
47 min
Osta 350,00 €

British military musket 1836+2 miekkaa
OULU, 90100
56 min
Huuda 249,00 €

KASA 1 KOTIMAINEN ISKELMÄ / POP
HELSINKI, 00840
33 min
6 huutajaa 22,50 €

Ilmakivääri Hatsan mod 125 ja gamo
HELSINKI, 00980
50 min
6 huutajaa 177,00 €

Vanha kirjoituspöytä
TURKU, 20750
31 min
Huuda 49,00 €

Suosituimmat
Näytä kaikki >

Tuubihuivi, vaalea
OULU, 90550
01.04.2016 02:45
Osta 8,50 €

Muumimuki Muumipeikko Unelmoi
MÄNTSÄLÄ, 04600
03.04.2016 20:00
8 huut. 1660,00 €

100 dollarin lompakko
POSTI, 00011
29.03.2016 23:55
Osta 7,90 €

Paavo Pesusieni bokserit XL-koko
YLIKIIMINKI, 91300
31.03.2016 13:37
Osta 9,90 €

Dual 606 levysoitin / Karjalan Antiikki
VANTAA, 01630
04.04.2016 12:26
16 huutajaa 81,95 €

Figure 6: Screenshot of the front page, taken 27.3.2016.

As can be seen from the figure, items on the front page are random and likely to be irrelevant for the users, especially for those who have the intention to buy only certain items. For example, a car and travel enthusiastic male user might not be interested in buying a stuffed wild boar or a woman's handbag as displayed on the front page in Figure 6.

This thesis aims to build a recommender system that makes the front page more relevant and personalized for the users. This is achieved by having the recommender system to choose the items to display on the front page. In this thesis, multiple recommenders are evaluated based on their predictive accuracy.

Several challenges, constraints and requirements are to be taken into account when building a recommender system. The following sections describe the used approaches, experimental setup, and the building process of potential recommenders.

4.2 Considerations

The large number of items and dynamic nature of the online auction are significant factors when considering recommendation techniques. There are continuously around 1.5 million items for sale on the auction site, and the average expiration time for them is around two weeks. Bidders may also close auctions by winning items before they expire. Moreover, tens of thousands of items are added on sale every day.

Nearly every item on the auction site is unique. Even items with the same product title often have dissimilarities, such as the condition, price, seller or the location of the item.

Even though hundreds of thousands users visit the site every week, many items have only few views, and some items may have no item views at all. The large number, uniqueness, and the dynamic nature of items lead to sparsity if represented in a traditional user-item rating matrix. Moreover, the dynamicity of items would require adding and removing item columns from the rating matrix columns continuously.

For practical considerations, the recommender system should be able to provide the recommendations quickly. The target requirement for computing the recommendation when it is queried is less than 100 milliseconds for the recommendation computation. This time limit allows for performing the necessary queries to display the recommendations, while not affecting the user experience through slow page loading times. The recommender should provide $N = 20$ recommendations to be displayed on the front page. The recommender should be able to recommend recently added items as well.

4.3 Data

The online auction site collects data about users and their activities on the website. The data available for this thesis are raw web log data of user clickstreams. In this data, users are identified based on their IP address. No data about users' previous events, such as sales, purchases or feedbacks exist. Moreover, detailed information about the auction items is not available.

Although the data available is limited, there is still vast amount of information to be utilized. The data allow for building and evaluating a recommender system. The data used in this thesis are divided into two different main datasets: *Dictionary dataset* and *Recommender dataset*, which are described in the following. However, due to data privacy and trade secrets, a very detailed exploratory analysis of the data is not provided.

1. Dictionary dataset

The Dictionary dataset is a one-year sample of user *item views* between June 2014 and May 2015. Item views are web log events where users have viewed auction items on their item-specific web pages. Each item view event in the web log includes information about the title of the item (*item title)* and the category of the item (*item category)*.

Item titles are parsed from the web logs, and they are missing special Finnish characters, such as "ä" and "ö". Non-alphabetical characters, and words or numbers with the length of one are removed from the titles. Because item titles are mostly product names, most of the words do not have inflected forms, and stemming is not considered necessary.

The online auction site uses a three-level item category hierarchy, where the first level has 22 categories, second level 95 categories, and the third level 737 categories. Item category column in this dataset represents the most specific, third level category of the item.

The Dictionary dataset has item views for over 4.7 million unique items which together cover all possible item categories. The dataset will be used later on as a dictionary and training data for content-based recommenders. A sample of the Dictionary dataset is shown in Table 5.

Table 5: A five row sample from the Dictionary dataset, where the columns are the row id, item category and item title.

Row id	**Item category**	**Item title**
1	417	*market black music placebo*
2	419	*europe youth megadeth nimmareilla juliste*
3	668	*hyvakuntoinen gloria heinakuu lehti*
4	586	*caprit marimekko ritva falla suunnittelemat*
5	96	*disney pentujengi alaskassa puhumme suomea*

2. Recommender dataset

The Recommender dataset contains all item views, *category views*, *category search queries* and *global search queries* from the auction site between June 2015 and November 2015. Category views are web log events where users have browsed items in a category page. Category search queries are events where users have performed search queries within a category page. Global search queries are events where users have performed searched queries from all categories. Each of these previously mentioned events includes an IP address based *user id*, *timestamp*, *session id* and *item id*, if it exists.

Here again, item titles are parsed from the web logs, and they are missing Finnish special characters. Moreover, any non-alphabetical characters and words or numbers of length one are removed from the titles. Once again, stemming is not considered necessary. In comparison to the Dictionary dataset, item categories can be any of the three category levels in the hierarchy.

The recommender dataset has hundreds of millions of events. For efficiency, the data is saved in a compressed format, where the event type can be inferred from empty values in different columns. Moreover, a new aggregated *words* –column is added to combine item titles and words used in search queries. Table 6 illustrates how event types can be inferred from empty values in columns.

Table 6: A sample of the Recommender dataset. The *inferred event type* column is not part of the real dataset, and in the table below, it indicates the event type of each row. For each row, the event type can be inferred from the empty cell values of the row.

Row id	**User id**	**Timestamp**	**Session id**	**Item category**	**Item id**	**Words (Search query / Item title)**	**Inferred event type**
1	36	2015-11-16 12:51:01	Rkjenio54	127	-	*walking dead*	Category search query
2	36	2015-11-19 14:52:55	Epka44df	215	383668	*walking dead figuuri rick*	Item view
3	1009	2015-08-16 08:15:01	4jk3l4aeds	-	-	*mcfarlane*	Global search query
4	23	2015-11-16 12:22:00	0f0fdkrY3	472	-	-	Category view

When the item views of Recommender dataset are represented in a user-item rating matrix, the sparsity is approximately 0.001 %, which is very sparse. Moreover, a large portion of the users in the dataset has only few sessions. The Recommender dataset is the main dataset used for recommender training and evaluation.

3. External item dataset

The External item dataset has around 4 million items with full item information. The closing times of these items span randomly mostly around November 2015. Only item titles and lowest level categories will be utilized from this dataset. The purpose of this dataset will be described in more detail in Section 4.7.1.

4.4 Implications on recommenders

The sparsity and dynamic nature of the rating matrix out rule traditional collaborative filtering techniques that use the rating matrix for computing recommendations. Moreover, sparsity in the matrix would lead to cold start problem, and some items would never get recommended since no users have experienced them. Since items are added and removed continuously on the auction site, the columns of the matrix should also be dynamically updated.

This problem could be approached by creating clusters, where items similar to each other belong to the same cluster based on some pre-defined item features. New items could be assigned to these clusters by calculating the distance to each cluster. However, in this thesis, the lack of extensive item data makes this difficult to implement. On the other hand, it is possible to use item categories as clusters, since items are similar to each other within categories. Categories also remain the same over time. Using categories as a starting point for recommendations in this sense is reasonable.

The lack of extensive item data affects the fine-tuning of the ultimate recommendation list. For example, two items with the same item title could be sorted by additional attributes, such as the price, auction type, seller or location. These additional attributes would also describe the implicit user preferences in more detail, such as the significance of different factors for the user.

To be able to recommend the most recently added items, part of the recommendation calculation has to be left for online computation, or at least the recommendations have to be updated frequently offline. To meet the recommendation time requirements, the recommendations should be pre-calculated offline as far as possible. In this thesis, a combination of offline and online calculation is used. User history is aggregated offline, but the actual recommendations are computed online, as close to user's query for recommendation as possible.

4.5 Recommendation process

Based on the requirements and implications described in the previous chapters, a general recommendation process is developed. The overview of this process is illustrated in Figure 7. It is a six step process that provides a general framework for producing the recommendation list from start to finish. The process is described in more detail in the following.

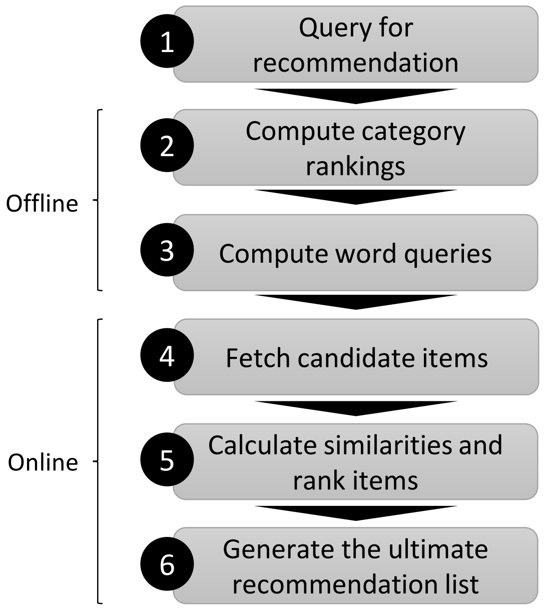

1
Query for recommendation
2
Compute category rankings
3
Compute word queries
4
Fetch candidate items
5
Calculate similarities and rank items
6
Generate the ultimate recommendation list
Offline
Online

Figure 7: The six step recommendation process.

1. **Query for recommendation**

In the first step, the recommender receives a recommendation query from the auction site to fetch top 20 items to be displayed for a certain user on the front page.

2. **Compute category rankings**

In the second step, categories of any level are selected by looping the most recent sessions of the user in chronological order. The categories can be of any category level in the category hierarchy. As long as there are more sessions to loop through, parameter $nSearches$ is not reached, and more than three sessions have not been looped through, categories are taken from the most recent sessions. $nSearches$ is a parameter that controls the number of *user word queries* that will be used in cosine similarity calculation (see formula (2.1)) later on. User word query refers to the word column in user's browsing history, and it can be either item title, category search query, or a global search query, as illustrated in Table 6.

Categories are then ranked in chronological order, where the most recent category is ranked the highest. Duplicate categories are removed by choosing the highest ranking for each of the duplicates. If the user has less than $nSearches$ user word queries available in the user history, and/or less than three experienced sessions, then all available categories are used. On the other hand, if the user has no previously experienced categories or word queries, the recommended items are selected randomly.

After ranking the categories obtained in the loop, similar categories to those are taken from pre-calculated category similarities (calculation described in Section 4.7.1). This is done by looping through the already ranked categories, and then fetching and ranking the similar categories after the already ranked categories. For example, if ten categories were obtained when looping through the most recent sessions of the user, the most similar category to the category that was ranked as #1 is ranked as #11, the most similar category to the category that was ranked as rank #2 is ranked as #12, and so on. Once again, duplicate categories are removed by choosing the highest ranking for each of the duplicates. The last step is to remove categories that are not of the lowest hierarchy level, in order to avoid fetching all candidate items from just one category later on. Because the pre-calculated category similarities has similarities between different hierarchy levels, it is possible to obtain ranking for all of the lowest level categories for the user. The obtained category rankings are used in later steps to fetch the candidate items.

3. Compute word queries

The third step is done in parallel with the second step. In the third step, word queries are selected from the most recent sessions in the same loop as for categories in step 2. As long as there are more sessions to loop through, parameter $nSearches$ is not reached, and more than three sessions have not been looped through, categories are taken from the most recent sessions in chronological order, and unique word queries are taken from the word column. Collecting user word queries this way guarantees that word queries chosen are relevant to the top ranked categories chosen. Similarly to step 2, if the user has less than $nSearches$ user word queries available in the session history, and/or less than three experienced sessions, then all available word queries are chosen.

At this point, category rankings and user word queries have been obtained. It is possible that more than $nSearches$ is collected when looping through the user's most recent events. Therefore, a simple heuristic is used to reduce the number of queries. First, cosine similarity is calculated between all the queries, where the vector space used is determined by the recommender. The end result is a similarity matrix where queries are in both the rows and the columns. Second, the column sums are calculated, and finally $nSearches$ number of queries with the smallest summed similarities are chosen. This heuristic aims to quickly choose the $nSearches$ number of distinct user word queries from all the available queries.

4. Fetch candidate items

In the fourth step, candidate items are fetched from the categories that were ranked in step 2. Items are chosen from the categories in the ranked order until $nCandItems$ number of items has been obtained from the categories. The parameter $nCandItems$ controls the maximum number of items chosen for similarity calculation against the user word queries from step 3. Items are assumed to be stored in an in-memory database, and the vector space is determined by the recommender.

5. Calculate similarities and rank items

In the fifth step, cosine similarities are calculated between the user word queries obtained in step 3 and the candidate items obtained in step 4. This results into a matrix of size $nSearches$ x $nCandItems$. After this, top $\mathrm{round}\left(\frac{nRecs}{nSearches-1}\right)$ number of most similar items are chosen for each query (row-wise). Candidate items with zero similarity are removed from each row. Parameter $nRecs$ controls the target number of ranked items for the recommendation list. For each query, the candidate items are sorted in descending order based on the similarity. The end result is a matrix, where each is row is sorted column-wise. This matrix is then transposed and flattened into a 1-dimensional array (2nd transposed row comes after 1st transposed row, 3rd after 2nd , and so on). Possible duplicate candidate items are removed from this array by keeping only the first occurrence for each. The resulting array represents the top items to be recommended for the user, in descending order based on the user word queries. Most of the consecutive items in the array are dissimilar, since they correspond to top items for different user word queries, and on the other hand, the queries were chosen to be distinct in step 3.

6. Generate the ultimate recommendation list

The sixth and final step is about generating the recommendation list. Instead of taking only the items from step 5, a simple modification is applied first. Based on initial experiments with different approaches, it was observed that users tend to visit the same items they have previously visited. Therefore, at most five of the most recently visited items by the user are chosen to be the first five items on the recommendation list. Thus, they are placed on ranks #1-5 on the recommendation list, depending whether the items are still available for sale. The recommendations computed in step 5 are added after the previously visited items to the recommendation list, with no modifications made in the ordering.

In the end, a ranked recommendation list is obtained. Maximum of five of the first five items are ones that the user has experienced before. Since items with zero similarity are removed from the recommendation list and only unique items are taken from the flattened array, it is possible that the number of ranked items in the end is more or less than $nRecs$. By controlling $nRecs$ in step 5, it is possible to guarantee a certain number of ranked items. However, increasing $nRecs$ requires more calculation and leads to slower recommendation times.

As shown in Figure 7, the category rankings and user word queries can be pre-calculated offline for each user as their user profile. Then, time will only be needed for fetching this pre-calculated user profile at the point of recommendation. The rest of the steps in the recommendation process have to be done online. The most time-consuming part is the similarity calculation in step 5, which could be parallelized if necessary. The time spent in online calculation can also be controlled via the previously mentioned parameters. The parameters that allow for controlling the recommendation process and calculation time are summarized in Table 7.

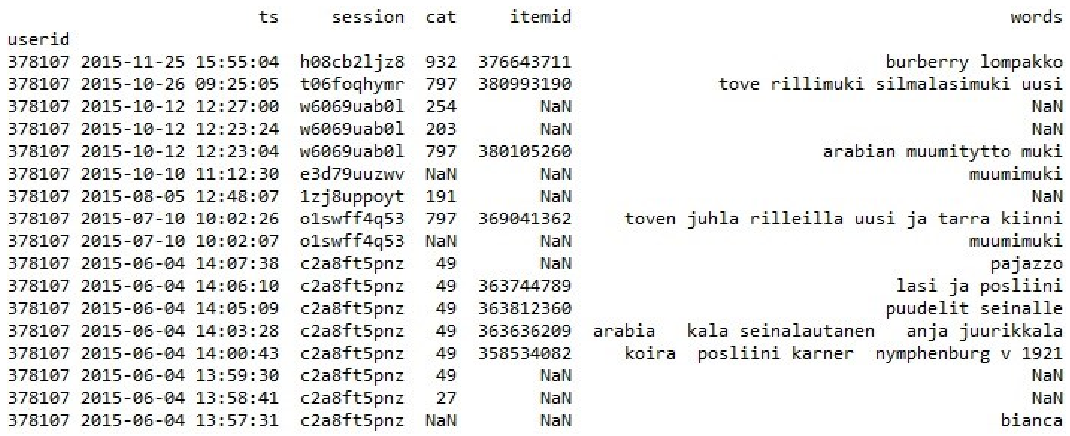
Table 7: Recommendation process parameters.

Parameter	Description
$nSearches$	The maximum number of user word queries that are taken from the browsing history of a user. The queries are used in similarity calculation against the candidate items. This parameter allows for controlling the similarity calculation time, and it can also be used for controlling the diversity of the output, since it allows for taking older word queries into account from the user's past.
$nCandItems$	The maximum number of items that are chosen from all possible items that are for sale. This parameter allows for controlling the similarity calculation time against the user word queries, and it directly controls the item coverage (see Section 3.2.2).
$nRecs$	The target number of ranked items that is needed from the recommender. It is time-efficient to not rank every possible item. However, it is not guaranteed that $nRecs$ number of items are ranked. If n_q is the number of items that needs to be guaranteed to be ranked, $nRecs$ should be chosen to be $n_q * (nSearches - 1)$.

4.6 Recommenders

A total of seven different recommenders are used in this thesis: *LastItems*, *Randomized*, *TFIDF*, *LSI*, *LSIub*, *LDAub* and *W2V*. LastItems and Randomized are used as reference point recommenders, and their recommendation process differs from the process described in Section 4.5 by not using the user word queries at all. Recommenders TFIDF, LSI, LSIub, LDAub and W2V obey the recommendation process and represent items in their respective vector spaces. They also use different kinds of training methods.

To illustrate the recommendations by each of these recommenders, a randomly chosen user browsing history is used as a case example. The browsing history of the chosen user is shown in Figure 8.



```
                        ts     session  cat     itemid                                           words
userid
378107 2015-11-25 15:55:04  h08cb2ljz8  932  376643711                               burberry lompakko
378107 2015-10-26 09:25:05  t06foqhymr  797  380993190                tove rillimuki silmalasimuki uusi
378107 2015-10-12 12:27:00  w6069uab0l  254        NaN                                             NaN
378107 2015-10-12 12:23:24  w6069uab0l  203        NaN                                             NaN
378107 2015-10-12 12:23:04  w6069uab0l  797  380105260                        arabian muumitytto muki
378107 2015-10-10 11:12:30  e3d79uuzwv  NaN        NaN                                      muumimuki
378107 2015-08-05 12:48:07  1zj8uppoyt  191        NaN                                             NaN
378107 2015-07-10 10:02:26  o1swff4q53  797  369041362      toven juhla rilleilla uusi ja tarra kiinni
378107 2015-07-10 10:02:07  o1swff4q53  NaN        NaN                                      muumimuki
378107 2015-06-04 14:07:38  c2a8ft5pnz   49        NaN                                         pajazzo
378107 2015-06-04 14:06:10  c2a8ft5pnz   49  363744789                               lasi ja posliini
378107 2015-06-04 14:05:09  c2a8ft5pnz   49  363812360                              puudelit seinalle
378107 2015-06-04 14:03:28  c2a8ft5pnz   49  363636209  arabia   kala seinalautanen   anja juurikkala
378107 2015-06-04 14:00:43  c2a8ft5pnz   49  358534082      koira  posliini karner  nymphenburg v 1921
378107 2015-06-04 13:59:30  c2a8ft5pnz   49        NaN                                             NaN
378107 2015-06-04 13:58:41  c2a8ft5pnz   27        NaN                                             NaN
378107 2015-06-04 13:57:31  c2a8ft5pnz  NaN        NaN                                          bianca
```

Figure 8: User history used as a case example to illustrate the output of different recommenders.

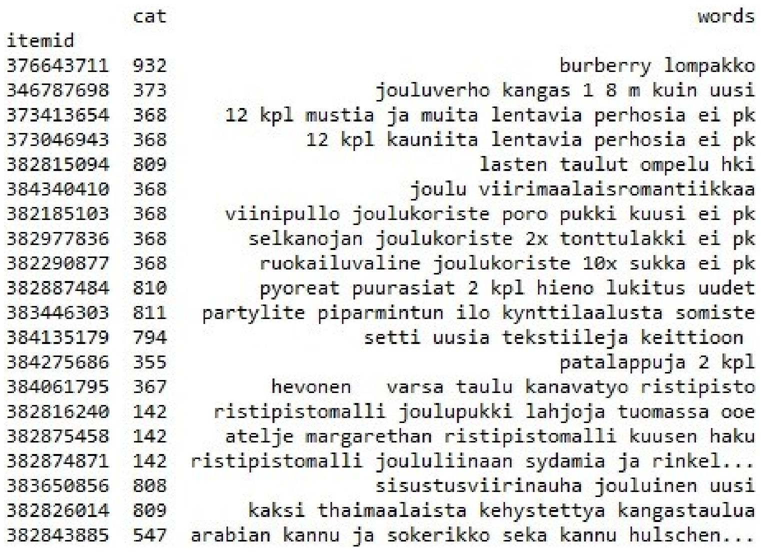
As it can be seen from the figure, the case example follows the format of the Recommender dataset as illustrated in Table 6. The user history consists of 7 unique sessions and categories visited over the past 6 months period, with a total of 12 user word queries available (*NaN* represents a missing value). This user history is used as an input for the recommenders to illustrate the output produced by different recommenders. Top 20 recommendations are produced by each of the recommenders. The available items for sale in the recommenders are the same that will be used in offline simulation later on (Section 4.7.1). Each recommender and the output for the case example is presented in the following sections.

4.6.1 Recommending previously visited items

Many users come back to see the same items they have previously visited. The recently visited items recommender, LastItems recommender, utilizes short-living items by ranking those items higher that the user has experienced in the past. The recommender ranks items based on their recency – the most recent item being ranked as #1, the second most recent as #2, and so on. If none of the previously visited items are still for sale, ranks are assigned randomly. This simple recommender is considered as a reference point recommender. The recommender avoids heavy computations, and allows for only storing maximum of N_u most recent items for each user. This is also the only parameter of the recommender.

The intuition behind the recommender is that users may want to come back to view the items they have previously considered interesting. For example, user might want to review the current bidding situation of an item. This simple recommender should already make the front page more relevant for the users, although it does not provide anything new to the users. However, since items are short-lived, users will not see the same items on the front page for a long time.

Figure 9 illustrates the output of this recommender for the case example in the ranked order. As it can be seen, from all the items visited by the case example user, only "*burberry lompakko"* is still available for sale, and thus it is ranked as #1. The rest of the items are chosen in the order they appear in the recommender, and they are most likely not relevant for the user.



```
           cat                                      words
itemid
376643711  932                          burberry lompakko
346787698  373             jouluverho kangas 1 8 m kuin uusi
373413654  368     12 kpl mustia ja muita lentavia perhosia ei pk
373046943  368           12 kpl kauniita lentavia perhosia ei pk
382815094  809                    lasten taulut ompelu hki
384340410  368               joulu viirimaalaisromantiikkaa
382185103  368     viinipullo joulukoriste poro pukki kuusi ei pk
382977836  368       selkanojan joulukoriste 2x tonttulakki ei pk
382290877  368        ruokailuvaline joulukoriste 10x sukka ei pk
382887484  810        pyoreat puurasiat 2 kpl hieno lukitus uudet
383446303  811   partylite piparmintun ilo kynttilaalusta somiste
384135179  794                setti uusia tekstiileja keittioon 
384275686  355                          patalappuja 2 kpl
384061795  367         hevonen   varsa taulu kanavatyo ristipisto
382816240  142    ristipistomalli joulupukki lahjoja tuomassa ooe
382875458  142     atelje margarethan ristipistomalli kuusen haku
382874871  142  ristipistomalli joululiinaan sydamia ja rinkel...
383650856  808             sisustusviirinauha jouluinen uusi
382826014  809       kaksi thaimaalaista kehystettya kangastaulua
382843885  547  arabian kannu ja sokerikko seka kannu hulschen...
```

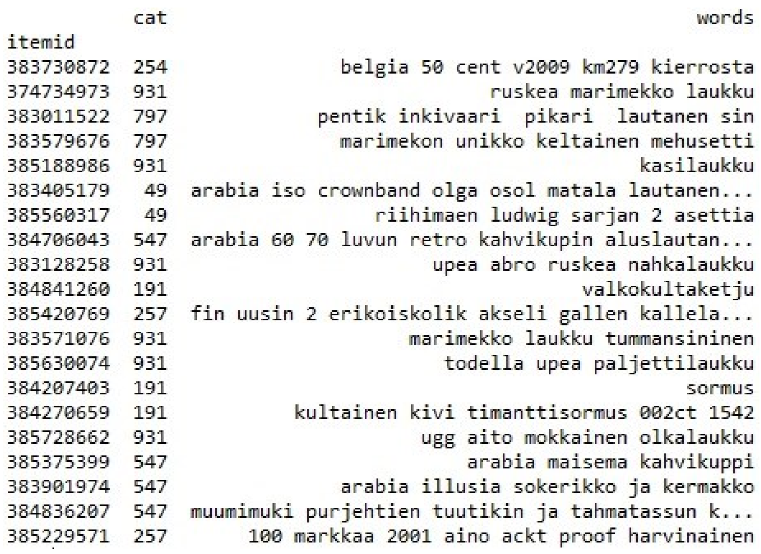
Figure 9: Top 20 recommendations for the case example by the LastItems recommender.

4.6.2 Recommending items from previously visited categories

The Randomized recommender ranks those items higher that are from the same lowest level categories that the user has experienced in the past. The intuition behind this recommender is that users are more likely to rank those items higher that are from the same categories they have previously experienced than those from other categories. This recommender ranks items in a certain category randomly, but only for the top N_{cat} categories, which are selected based on the second step of the recommendation process (compute category rankings, as described in Section 4.5). The parameters for this recommender are N_{cat} and $nSearches$, where the latter controls the ranking of the categories.

The Randomized recommender is also considered as a baseline model to which compare other recommenders to. It provides a simple heuristic for making the displayed items more relevant to the users.

Figure 10 illustrates recommendations by the Randomized recommender for the case example. Because previously visited items are only taken into account broadly only through their category, users may find new items that are similar to previously visited items. Since no user word queries are used to produce the recommendations, items from those categories are also recommended where the user has only visited but not viewed an item. For example, in this case example, user has visited category 191, but did not view any item or perform a search query in that category. Even so, items from category 191 are recommended.



```
           cat                                           words
itemid
383730872  254              belgia 50 cent v2009 km279 kierrosta
374734973  931                          ruskea marimekko laukku
383011522  797             pentik inkivaari  pikari  lautanen sin
383579676  797              marimekon unikko keltainen mehusetti
385188986  931                                       kasilaukku
383405179   49  arabia iso crownband olga osol matala lautanen...
385560317   49                 riihimaen ludwig sarjan 2 asettia
384706043  547  arabia 60 70 luvun retro kahvikupin aluslautan...
383128258  931                        upea abro ruskea nahkalaukku
384841260  191                                   valkokultaketju
385420769  257  fin uusin 2 erikoiskolik akseli gallen kallela...
383571076  931                      marimekko laukku tummansininen
385630074  931                         todella upea paljettilaukku
384207403  191                                            sormus
384270659  191           kultainen kivi timanttisormus 002ct 1542
385728662  931                        ugg aito mokkainen olkalaukku
385375399  547                           arabia maisema kahvikuppi
383901974  547              arabia illusia sokerikko ja kermakko
384836207  547  muumimuki purjehtien tuutikin ja tahmatassun k...
385229571  257       100 markkaa 2001 aino ackt proof harvinainen
```

Figure 10: Top 20 recommendations for the case example by the Randomized recommender.

4.6.3 Term Frequency-Inverse Document Frequency

The *Term Frequency-Inverse Document Frequency* (TFIDF) recommender represents items in TFIDF vector space, where those items are ranked higher that have similar item titles to those items that the user has experienced in the past. Adopting from Lops et al. (2011), the general TFIDF vector space model is represented in the context of this thesis (Lops, et al., 2011).

In TFIDF every item can be represented as a vector of term weights, where each weight indicates the degree of association between the item and the term. Let $\mathcal{I}_t = \{i_i, i_2, \dots, i_N\}$ be a set of item titles and $T = \{t_1, t_2, \dots, t_n\}$ be the set of words in the set of item titles. Now, by representing each item i_k as a vector in d_{TFIDF} dimensional space, it is possible to obtain $i_k = \{w_{1k}, w_{2k}, \dots, w_{nk}\}$, where w_{jk} is a weight for term t_j in item i_k.

In TFIDF, those terms that occur frequently in one item title but rarely in other item titles, are more likely to be relevant to that item title. The weights can be written as follows

$$w_{j,k} = TF(t_j, i_k) \times \log_2\left(\frac{N}{n_j}\right), \tag{4.11}$$

where N is the number of items in the set of items $\mathcal{I}_t$, and n_j is the number of items in the set of items in which the term t_j occurs at least once. Here, the term frequency is

$$TF(t_j, i_k) = \frac{f_{j,k}}{\max_z f_{z,k}}, \tag{4.12}$$

where the maximum is computed over the frequencies $f_{z,k}$ of all terms t_z that occur in item title i_k. Now, with normalization to unit length the weights can be represented as

$$w_{j,k} = \frac{f_{j,k}}{\max_z f_{z,k}} \times \log_2\left(\frac{N}{n_j}\right), \tag{4.13}$$

which means that each item i_k can be represented as a vector in d_{TFIDF} dimensional space. By following formula (2.1), the similarity of two items can be computed with cosine similarity

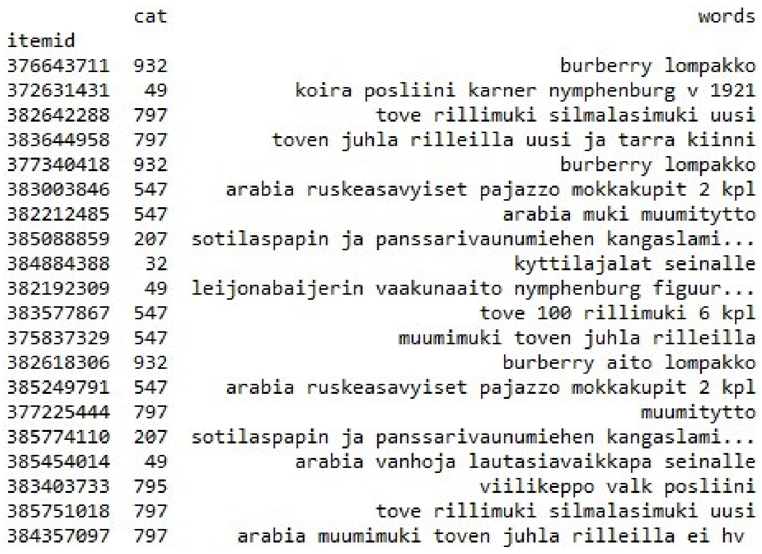
$$sim(i_a, i_b) = \frac{\sum_j w_{j,a} \cdot w_{j,b}}{\sqrt{\sum_j w_{j,a}^2} \cdot \sqrt{\sum_j w_{j,b}^2}}. \tag{4.14}$$

TFIDF recommender follows the general recommendation process as described in Section 4.5, and in TFIDF vector space. TFIDF recommender allows for representing both the user browsing history (experienced items) and the items for sale in the same d_{TFIDF}-dimensional space for quick online similarity computation.

The computation time is mostly spent on the similarity calculation, where the time depends on the number of user word queries used, and the number of candidate items chosen. However, item titles are short, and the number of terms is smaller than the dimension d_{TFIDF}, which allows for sparse matrix representation.

The parameters for TFIDF recommender are the dimension d_{TFIDF}, which can be controlled by choosing the number of most frequent terms to keep. The recommendation process parameters $nSearches$, $nCandItems$ and $nRecs$ are also parameters of the TFIDF recommender.

Figure 11 illustrates recommendations by the TFIDF recommender for the case example. Since recommendations are based on the words as is, each recommendation must contain words that are already present in the user search queries. For example, words "*arabia*" and "*burberry*" are included in many of the recommendations.



```
           cat                                              words
itemid
376643711  932                                  burberry lompakko
372631431   49             koira posliini karner nymphenburg v 1921
382642288  797                      tove rillimuki silmalasimuki uusi
383644958  797            toven juhla rilleilla uusi ja tarra kiinni
377340418  932                                  burberry lompakko
383003846  547        arabia ruskeasavyiset pajazzo mokkakupit 2 kpl
382212485  547                            arabia muki muumitytto
385088859  207   sotilaspapin ja panssarivaunumiehen kangaslami...
384884388   32                              kyttilajalat seinalle
382192309   49   leijonabaijerin vaakunaaito nymphenburg figuur...
383577867  547                             tove 100 rillimuki 6 kpl
375837329  547                      muumimuki toven juhla rilleilla
382618306  932                             burberry aito lompakko
385249791  547        arabia ruskeasavyiset pajazzo mokkakupit 2 kpl
377225444  797                                        muumitytto
385774110  207   sotilaspapin ja panssarivaunumiehen kangaslami...
385454014   49             arabia vanhoja lautasiavaikkapa seinalle
383403733  795                            viilikeppo valk posliini
385751018  797                      tove rillimuki silmalasimuki uusi
384357097  797         arabia muumimuki toven juhla rilleilla ei hv
```

Figure 11: Top 20 recommendations for the case example by the TFIDF recommender.

4.6.4 Latent Semantic Indexing

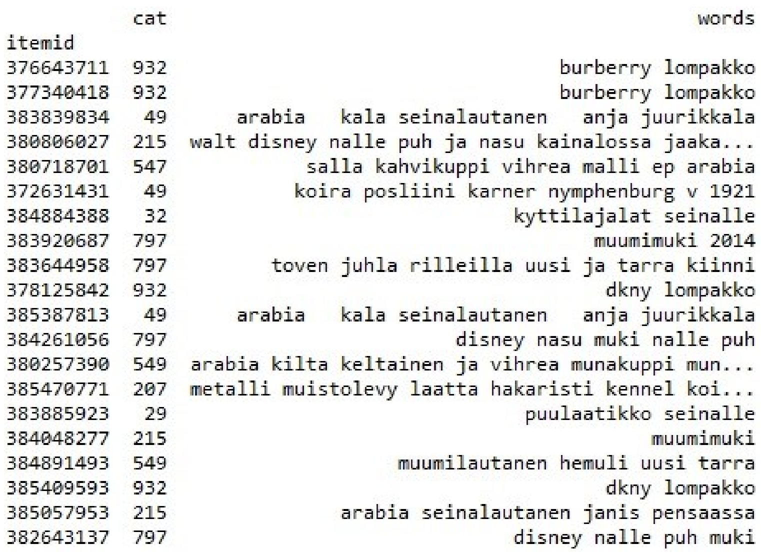
Latent Semantic Indexing (LSI), also known as *Latent Semantic Analysis* (LSA), was proposed by Deerwester et al. in 1990 (Deerwester, et al., 1990). It is a theory and method that extracts and represents contextual-usage meaning of words by statistical computations applied to a text corpus. It determines the similarity of meaning of words and sets of words by aggregating all the word contexts in which a given word does and does not appear (Landauer, et al., 1998).

LSI applies a *singular value decomposition* (SVD) to a document-term matrix, where unique term weights are in the rows and documents are in the columns. SVD decomposes this matrix into a product of three other matrices. One component matrix describes the original row entities as vectors of derived orthogonal factor values, another describes the original column entities in the same way, and the third is a diagonal matrix containing scaling values. The original matrix can be reconstructed by multiplying these three matrices, and SVD allows for dimension reduction by ordinarily deleting coefficients from the diagonal matrix, starting from the smallest (Landauer, et al., 1998).

In this thesis, item titles are considered as documents. The LSI recommender transforms items from TFIDF vector space to LSI space by reducing the dimension of TFIDF weighted term matrix from d_{TFIDF} to d_{LSI}. The highest valued coefficients of the diagonal matrix are used for transforming new, unseen items into the LSI space.

The parameters for LSI recommender include all the parameters from TFIDF recommender, such as the dimension d_{TFIDF} and recommendation process parameters $arches$, $nCandItems$, and $nRecs$. Moreover, parameter d_{LSI} controls the target LSI-dimension.

Figure 12 illustrates recommendations by the LSI recommender for the case example. Since LSI aggregates all the word contexts in which a given word does and does not appear, it is possible to have items on the recommendation list that contains no words or only few words from the original user word queries. For example, *"walt disney nalle puh ja nasu kainalossa jaakaappimagneetti"* is fourth on the list, even though *"ja"* is a common Finnish word and it is the only word that appears in the user word query. The high ranking could be due to similarities between words *"muki"* and *"disney*", *"nalle"* or *"puh".* These word similarities are learned by the recommender in advance, during the *training* process of the recommender. The training of each recommender is described on a high-level in Section 4.7.2.



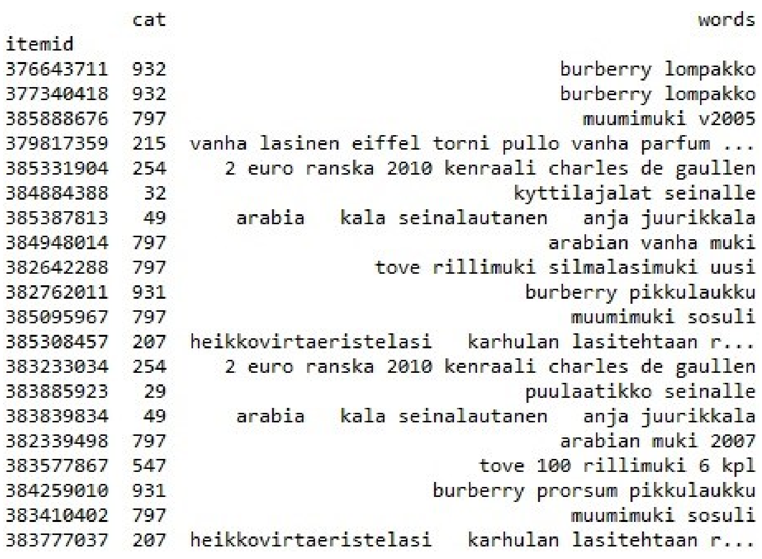
```
           cat                                            words
itemid
376643711  932                                burberry lompakko
377340418  932                                burberry lompakko
383839834   49      arabia   kala seinalautanen   anja juurikkala
380806027  215  walt disney nalle puh ja nasu kainalossa jaaka...
380718701  547              salla kahvikuppi vihrea malli ep arabia
372631431   49             koira posliini karner nymphenburg v 1921
384884388   32                            kyttilajalat seinalle
383920687  797                                     muumimuki 2014
383644958  797         toven juhla rilleilla uusi ja tarra kiinni
378125842  932                                     dkny lompakko
385387813   49      arabia   kala seinalautanen   anja juurikkala
384261056  797                          disney nasu muki nalle puh
380257390  549  arabia kilta keltainen ja vihrea munakuppi mun...
385470771  207  metalli muistolevy laatta hakaristi kennel koi...
383885923   29                              puulaatikko seinalle
384048277  215                                         muumimuki
384891493  549                    muumilautanen hemuli uusi tarra
385409593  932                                     dkny lompakko
385057953  215               arabia seinalautanen janis pensaassa
382643137  797                               disney nalle puh muki
```

Figure 12: Top 20 recommendations for the case example by the LSI recommender.

4.6.5 User-based Latent Semantic Indexing

User-based Latent Semantic Indexing (LSIub) recommender is similar to LSI recommender, and the only difference is in the training process. In the training process of LSIub, new word contexts are artificially created by joining the words of user sessions in a training corpus. The intuition is to show the recommender which words have tendency to appear together within a single browsing session, and use these as individual training documents. The parameters for LSIub are similar to LSI, including $d_{LSI}, d_{TFIDF}, nSearches, nCandItems$ and $nRecs$.

Figure 13 illustrates recommendations by the LSIub recommender for the case example. Since LSIub not only aggregates all the word contexts in which a given word does and does not appear, but also creates new word contexts by aggregating words from the sessions of different users in the past, it is possible to have recommendations that are significantly different from the original user word queries. For example, LSIub recommends a 2€ coin *“2 euro ranska 2010 kenraali charles de* *gaullen"* for the case user, even though only the category 254 view indicates that the user could be interested in coins. However, the new artificially created contexts might have had examples where users have experienced both *"arabia"/"muumi"* and coin related items during the same session. This would be reasonable since all the previously mentioned words are popular among collectors.



```
           cat                                               words
itemid
376643711  932                                   burberry lompakko
377340418  932                                   burberry lompakko
385888676  797                                     muumimuki v2005
379817359  215  vanha lasinen eiffel torni pullo vanha parfum ...
385331904  254     2 euro ranska 2010 kenraali charles de gaullen
384884388   32                               kyttilajalat seinalle
385387813   49       arabia   kala seinalautanen   anja juurikkala
384948014  797                                  arabian vanha muki
382642288  797                   tove rillimuki silmalasimuki uusi
382762011  931                               burberry pikkulaukku
385095967  797                                    muumimuki sosuli
385308457  207  heikkovirtaeristelasi   karhulan lasitehtaan r...
383233034  254     2 euro ranska 2010 kenraali charles de gaullen
383885923   29                               puulaatikko seinalle
383839834   49       arabia   kala seinalautanen   anja juurikkala
382339498  797                                   arabian muki 2007
383577867  547                             tove 100 rillimuki 6 kpl
384259010  931                         burberry prorsum pikkulaukku
383410402  797                                    muumimuki sosuli
383777037  207  heikkovirtaeristelasi   karhulan lasitehtaan r...
```

Figure 13: Top 20 recommendations for the case example by the LSIub recommender.

4.6.6 User-based Latent Dirichlet Allocation

In general, *Latent Dirichlet Allocation* (LDA) is a generative probabilistic topic model introduced by Blei et al. in 2003. The underlying idea in LDA is that documents are represented as random mixtures over latent topics k, where topics are characterized by a distribution over words. LDA assumes that each document in a collection of documents is generated through the following process (Blei, et al., 2003):

1. Choose a number of words $N \sim Poisson(\xi)$ from the document, where $Poisson$ is the Poisson distribution.
2. Generate d_{LDA} topics by randomly choosing a word distribution $\theta_k \sim Dir(\beta)$ for each topic $k \in K$, where Dir is the Dirichlet distribution.
3. For each of the N words w_i:

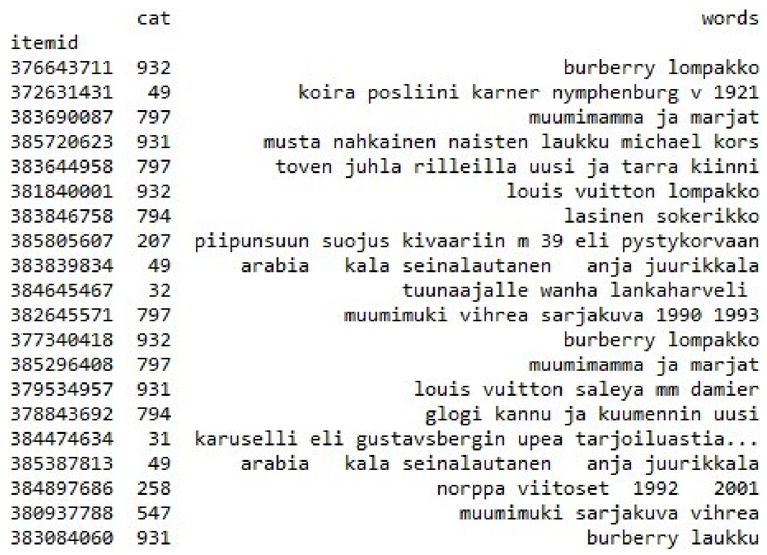
a. Choose a topic $z_i \sim Multinomial(\theta)$, where $Multinomial$ is the multinomial distribution.
b. Choose a word w_i from $P(w_i|z_i, \beta)$, which is a multinomial probability conditioned on topic z_i.

Because distributions for the model parameters are difficult to compute directly, approximate inference algorithms have been proposed, such as *variational inference* (Blei, et al., 2003) or *Gibbs sampling* (Griffiths & Steyvers, 2004).

In LDA vector space, items are represented as d_{LDA} dimensional vectors, where the vector elements are the latent topics. Similar to LSIub recommender, LDAub recommender is trained on documents, where each document consists of the words that appear within a single browsing session. In early experimentation with LDA it was observed that LDA solely trained on individual words leads to poor accuracy due to short length of individual item titles. Therefore, only LDAub is considered in this thesis.

LDAub follows the previously presented recommendation process with parameters $arches$, $nCandItems$, and $nRecs$. In addition, the number of latent topics d_{LDA} is a parameter of the recommender.

Figure 14 illustrates recommendations by the LDAub recommender for the case example. Since recommendations in LDAub are based on similarities of latent topics rather than exact words, many items are recommended even when the exact words do not appear in the original user word queries. Some of the recommended items, such as gun related *"piipunsuun suojus kivaariin m 39 eli pystykorvaan"* or handcraft related *"tuunaajalle wanha lankaharveli"*, would seem to be irrelevant based on the user's history.



```
           cat                                          words
itemid
376643711  932                               burberry lompakko
372631431   49          koira posliini karner nymphenburg v 1921
383690087  797                             muumimamma ja marjat
385720623  931       musta nahkainen naisten laukku michael kors
383644958  797        toven juhla rilleilla uusi ja tarra kiinni
381840001  932                          louis vuitton lompakko
383846758  794                               lasinen sokerikko
385805607  207  piipunsuun suojus kivaariin m 39 eli pystykorvaan
383839834   49      arabia   kala seinalautanen   anja juurikkala
384645467   32                  tuunaajalle wanha lankaharveli 
382645571  797             muumimuki vihrea sarjakuva 1990 1993
377340418  932                               burberry lompakko
385296408  797                             muumimamma ja marjat
379534957  931                    louis vuitton saleya mm damier
378843692  794                     glogi kannu ja kuumennin uusi
384474634   31  karuselli eli gustavsbergin upea tarjoiluastia...
385387813   49      arabia   kala seinalautanen   anja juurikkala
384897686  258                        norppa viitoset  1992   2001
380937788  547                         muumimuki sarjakuva vihrea
383084060  931                                  burberry laukku
```

Figure 14: Top 20 recommendations for the case example by the LDAub recommender.

4.6.7 Word2Vector

In 2013, Mikolov et al. proposed two novel model architectures for computing continuous vector representations of words for large datasets (Mikolov, et al., 2013). Their *Word2Vec* (W2V) model learns vector representations for words by using a shallow neural network language model. One of these is the skip-gram neural network architecture that consists of an input layer, a projection layer and an output layer to predict nearby words. Due to its simple architecture, the skip-gram model can be trained with billions of words per hour on a single conventional desktop computer. Word vectors are trained to maximize log probability of neighboring words in a given sequence of words $w_1, w_2, \dots, w_N$

$$\frac{1}{N}\sum_{t=1}^{N}\sum_{j\in nb(t)}\log P(w_j|w_t), \tag{4.15}$$

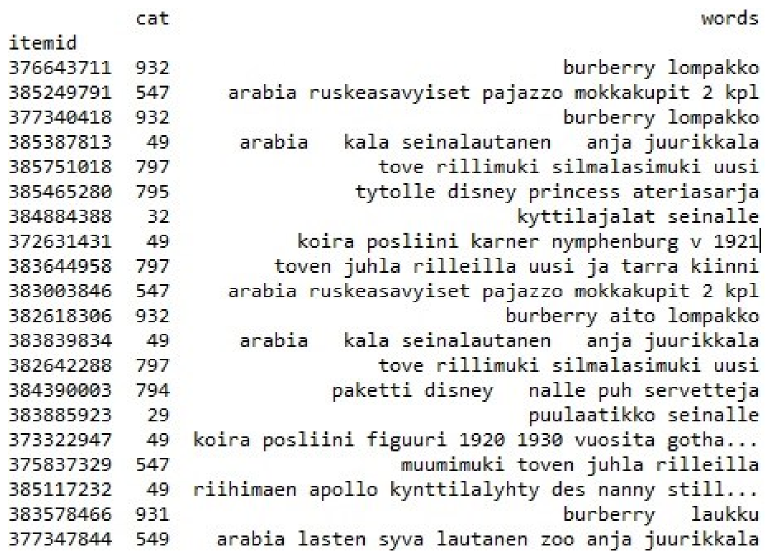
where $nb(t)$ is the set of neighboring words of w_t, and $P(w_j|w_t)$ is a hierarchical softmax of the associated word vectors v_{w_j} and v_{w_i}. The model can learn complex word relationships, such as $vec(Japan) - vec(Sushi) + vec(Germany) \approx vec(bratwurst)$ (Kusner, et al., 2015; Mikolov, et al., 2013).

The W2V recommender utilizes W2V model's ability to learn associations from neighboring words by combining words from users' sessions, similar to LSIub and LDAub. The intuition is to show W2V model which words tend to appear together within a browsing session, aiming to include this information in the vector representations of words. For example, if two words appear within the same browsing session often, their vectors should be more similar than of those words which never appear within a session.

Since each word is represented by a vector, user word queries and item titles that consist of multiple words need to be aggregated. To tackle this problem, different techniques have been proposed, such as *Paragraph2Vec* (Mikolov & Le, 2014). These techniques take the word order into account, but in the case of item titles and user word queries, the word order is not crucial. During initial experiments with W2V recommender, it was found that decent predictive accuracy could be achieved by taking the mean of the word vectors and using cosine similarity to calculate similarities in the W2V vector space.

The parameters for W2V are the recommendation process parameters $nSearches, nCandItems$ and $nRecs$. Moreover, parameter d_{W2V} controls the word vector representation dimension, and $window$ parameter controls the maximum distance to be considered as the word neighborhood within a sentence.

Figure 15 illustrates recommendations by the W2V recommender for the case example. Since recommendations in W2V are based on neighboring words and user sessions, some items are recommended even though the exact words do not appear in the original user word queries, such as *"paketti disney nalle puh servetteja"*. This behavior is similar to LSIub and LDAub. On the other hand, nearly all the words that appear in the query also appear in the recommendation list as in TFIDF.



```
           cat                                            words
itemid
376643711  932                               burberry lompakko
385249791  547     arabia ruskeasavyiset pajazzo mokkakupit 2 kpl
377340418  932                               burberry lompakko
385387813   49      arabia   kala seinalautanen   anja juurikkala
385751018  797                 tove rillimuki silmalasimuki uusi
385465280  795                tytolle disney princess ateriasarja
384884388   32                           kyttilajalat seinalle
372631431   49          koira posliini karner nymphenburg v 1921
383644958  797          toven juhla rilleilla uusi ja tarra kiinni
383003846  547     arabia ruskeasavyiset pajazzo mokkakupit 2 kpl
382618306  932                          burberry aito lompakko
383839834   49      arabia   kala seinalautanen   anja juurikkala
382642288  797                 tove rillimuki silmalasimuki uusi
384390003  794               paketti disney   nalle puh servetteja
383885923   29                            puulaatikko seinalle
373322947   49  koira posliini figuuri 1920 1930 vuosita gotha...
375837329  547                  muumimuki toven juhla rilleilla
385117232   49  riihimaen apollo kynttilalyhty des nanny still...
383578466  931                               burberry   laukku
377347844  549    arabia lasten syva lautanen zoo anja juurikkala
```

Figure 15: Top 20 recommendations for the case example by the W2V recommender.

4.7 Experimental setup

The following describes the experimental setup that is used for simulation and evaluation of the recommenders. First, it is described how the available data is preprocessed and split into smaller datasets. Second, the simulation setup and training of the recommenders are described. Third, the evaluation metrics are introduced.

4.7.1 Data preprocessing

The three datasets introduced in Section 4.3 were the Dictionary dataset, Recommender dataset and External Item dataset. Figure 16 illustrates how the Recommender dataset and External Item dataset are processed into smaller datasets for simulation purposes.

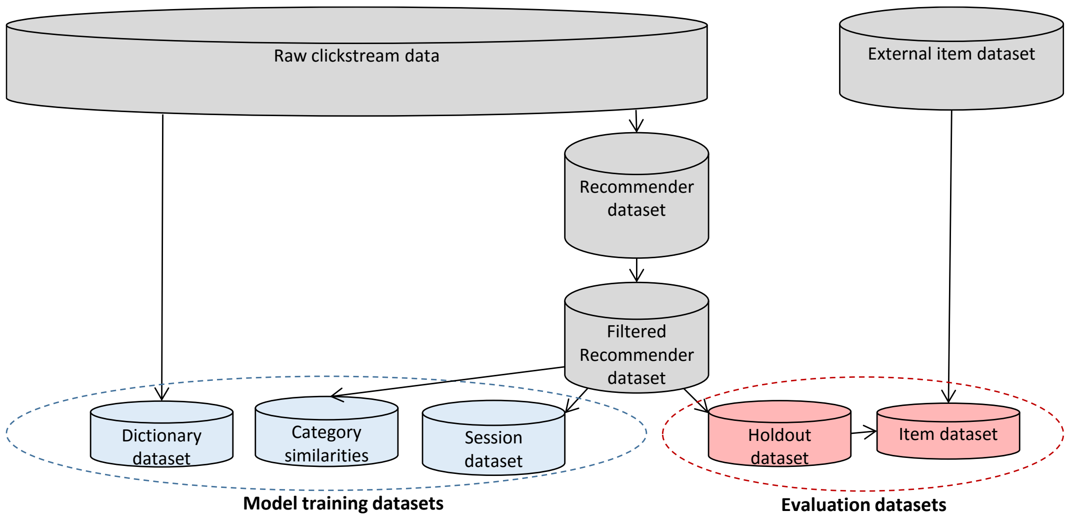


Figure 16: Datasets used in the experimental setup.

The Recommender dataset is filtered to only contain users that have item views between dates 15.11.2015 – 30.11.2015. Moreover, only those users are chosen who have at least 6 sessions (1 session / month), and at most 1800 sessions (10 sessions / day) between the six month period 1.6.2015 – 30.11.2015. This dataset is called the *Filtered Recommender dataset*, still including tens of millions of rows, and the same five columns as the original Recommender dataset: user id, timestamp, session id, item category, item id and words.

The Filtered Recommender dataset is further on processed into *Session dataset*. The Session dataset is the Filtered Recommender dataset that is filtered by dates between 1.6.2015 – 14.11.2015. This data is then aggregated by sessions so that all the words that appear within a session are joined together with white spaces. The Session dataset is used for training the user-based models LSIub, LDAub and W2V. This dataset has several million sessions, and since the session id's are not needed, the dataset is only a list of the joined words.

The Filtered Recommender dataset is also processed into *Category Similarities*. Category similarities are calculated by first filtering the Filtered Recommender dataset to only contain data from dates between 1.6.2015 – 1.11.2015. Then, a sparse session id – category id binary rating matrix is constructed. In this matrix, sessions are in the rows and categories in the columns. If a certain category was visited in a session, the value of the category-session pair is 1, and 0 otherwise. The categories may be of any level in the category hierarchy. Finally, cosine similarity between the categories is calculated, and the results are sorted to have the most similar categories to a certain category in descending order in the columns. The similarity of a category to itself is ranked as last. In the end, the Category Similarities contain the most similar categories for all existing categories, and across all category levels.

The *Holdout dataset* is used for recommender evaluation, and it is obtained by filtering dates 20.11.2015 – 27.11.2015 from the Filtered Recommender dataset. The data from these dates is not seen beforehand by any of the recommenders, which allows for recommender evaluation on this dataset.

The *Item dataset* is combined from External Item dataset and Holdout dataset. This dataset approximates the items that are available for sale during the evaluation period 20.11.2015 – 27.11.2015. It is based on the assumption that the items available on the auction site between dates 20.11.2015 – 27.11.2015 stay relatively constant. With this assumption, it is possible to collect item titles from all unique item views in the Holdout dataset, and combine these with items in the External Item dataset. The External Item dataset is filtered to consist of items that have listing date before 20.11.2015 and closing date after 27.11.2015. By combining these datasets, a total of $1\ 354\ 476$ items with their item title and category are obtained. The total number of items is close to the average number of 1.5 million items that are for sale on the auction site.

4.7.2 Simulation

The recommenders are evaluated offline by simulating them in a real life scenario for one week time period 20.11.2015 – 27.11.2015. All the recommenders are implemented in Python, with the help of topic modelling library *gensim* by Radim Řehůřek (Řehůřek, 2016). The simulation steps are described in the following.

First, the recommenders are trained. The exact training process depends on the recommender, and more information about the training process of each vector space model can be found in their corresponding literature. However, the datasets used for training are described here. Terms and term weights for TFIDF, LSI, LSIub and LDAub recommenders are trained using the Dictionary dataset. Words that appear less than 5 times are removed from the vocabulary, and only the 200 000 most frequent words are kept in the dictionary. Terms for W2V are taken from the words that appear in the Session dataset, and only the words that appear more than 20 times are kept in W2V. User based recommenders LSIub, LDAub and W2V are also trained on the Session dataset, using the terms trained in each of the recommenders. For TFIDF and LSI, only the Dictionary dataset is needed in the training phase. LastItems and Randomized recommenders use only the user history data, and they do not require training.

The parameters for each recommender are chosen based on initial experiments with the recommenders. In these experiments, the prediction accuracy, coverage, diversity, and time spent in producing the recommendations are taken into account. The parameters chosen are presented in Table 8. Because the recommendations are made for the front page, $nRecs = 100$ are chosen as target for the VSM recommenders. Choosing $nCandItems = 80\ 000$ to limit the possible items to recommend results in item space coverage of $SISC = \frac{80\ 000}{1\ 354\ 476} \approx 5.91\ \%$.

Table 8: Parameters chosen for each recommender.

Recommender	Parameter
LastItems	• $N_u = 100$
Randomized	• $N_{cat} = 10$ • $nSearches = 8$
TFIDF	• $d_{TFIDF} = 200\,000$ • $nSearches = 8$ • $nCandItems = 80\,000$ • $nRecs = 100$
LSI	• $d_{LSI} = 100$ • $d_{TFIDF} = 200\,000$ • $nSearches = 8$ • $nCandItems = 80\,000$ • $nRecs = 100$
LSIub	• $d_{LSI} = 100$ • $d_{TFIDF} = 200\,000$ • $nSearches = 8$ • $nCandItems = 80\,000$ • $nRecs = 100$
LDAub	• $d_{LDA} = 100$ • $nSearches = 8$ • $nCandItems = 80\,000$ • $nRecs = 100$
W2V	• $d_{W2V} = 100$ • $window = 5$ • $nSearches = 8$ • $nCandItems = 80\,000$ • $nRecs = 100$

Second, samples for evaluation are chosen from the Holdout dataset. To keep the user activity balanced in the sample, a stratified sample of $14\ 218$ users is selected from the Holdout set. The stratification is based on binning the users into the following session count quantiles: $[0,10]$, $(10,25]$, $(25,50]$, $(50,75]$, $(75,90]$, and $(90,100]$. This ensures that both more and less active users are chosen for evaluation. The chosen sample has a total of $153\ 847$ item views to be predicted during the simulation time period.

Third, each recommender is evaluated by making recommendations for each user in the sample, one day at a time. To simulate real life scenario, the user history is updated to have the latest information about the users' activities from the previous day. For example, on 25.11.2016, the recommenders are able to see the users' activities from 24.11.2016 and before. The user information is assumed to be updated every night when the day changes, with no additional update delays. The evaluation metrics for each recommendation made are computed and stored, and they are described in the following.

4.7.3 Evaluation

As described in Chapter 3, recommenders can be evaluated with respect to many different criteria. In this thesis, user ratings to be predicted are implicit and binary (whether the user experienced the item or not). By assuming that correctly predicted items would make the site more relevant, it is reasonable to measure prediction accuracy and the ranking of items overall.

Since the number of recommended items is around 100 for most of the recommenders (see Table 8), it is reasonable to measure True-Positive-Rate, precision, and F1-Score at different recommendation list sizes $N < 100$. The sizes chosen are $N = 1, 5, 10, 15, 20, 25, 30, 40, 60, 80$ and 100.

To examine how well the recommenders can recommend new items to users, the previously mentioned metrics are also computed by using only those items that the users have not seen before. The metrics computed using all items are called *visited items*, and the metrics calculated with new items only are called *non-visited items*. As mentioned in Section 3.2.1, it should be noted that the number of False-Positives is likely to be overestimated in offline evaluation. Therefore, the recommenders may seem to perform worse in comparison to if they were actually implemented at the time of evaluation. After simulation, the metrics are aggregated for all recommendation list sizes and for both visited and non-visited items.

Because predicting a small number of items out of a large collection of items is difficult, it makes sense to measure how high items in the holdout set are ranked by the recommender. Therefore, NDCG and R-Score with $\alpha = 20$ (R20) and 100 (R100) are chosen for measuring the ranking of items. The items not ranked by the recommender due to $nRecs$ limitation are all set to have rank of the last possible rank position, which is the number of items in the recommender. The ranking metrics are calculated for visited items only.

The recommenders are also evaluated with respect to *aggregation time* and *prediction time*. Aggregation time is the time taken by steps 2-4 in the recommendation process (Figure 7), whereas prediction time is the time taken by steps 5-6 in the same figure.

5 Results

In order to obtain an overview of the performance of each recommender, the results are visualized in the following. The exact numerical results are shown in Appendix A – Table of evaluation results.

The True-Positive-Rate for visited and non-visited items at different recommendation list sizes is shown in Figure 17. The figure shows that there are three recommender groups that have similar performance: TFIDF/W2V, LSI/LSIub, and LDAub/LastItems. TFIDF and W2V are the best performers for all recommendation list sizes, and for both visited and non-visited items. Thus, the proportion of correctly predicted items from all items that users have experienced is on average the highest for these recommenders. For both non-visited and visited items, and at all recommendation list sizes, LSI and LSIub perform better than LDAub and LastItems but worse than TFIDF and W2V. For visited items, all the previously mentioned recommenders perform better than the Randomized recommender, whose performance is clearly the worst.

As expected, TPR for the non-visited items is zero for LastItems recommender, since it does not recommend unseen items. Moreover, the non-visited item curve for all recommenders show a gentle slope because five of the most recently experienced items are ranked in the top five, assuming that they are still available for sale. Overall, TPR is lower for non-visited items than for the visited items.

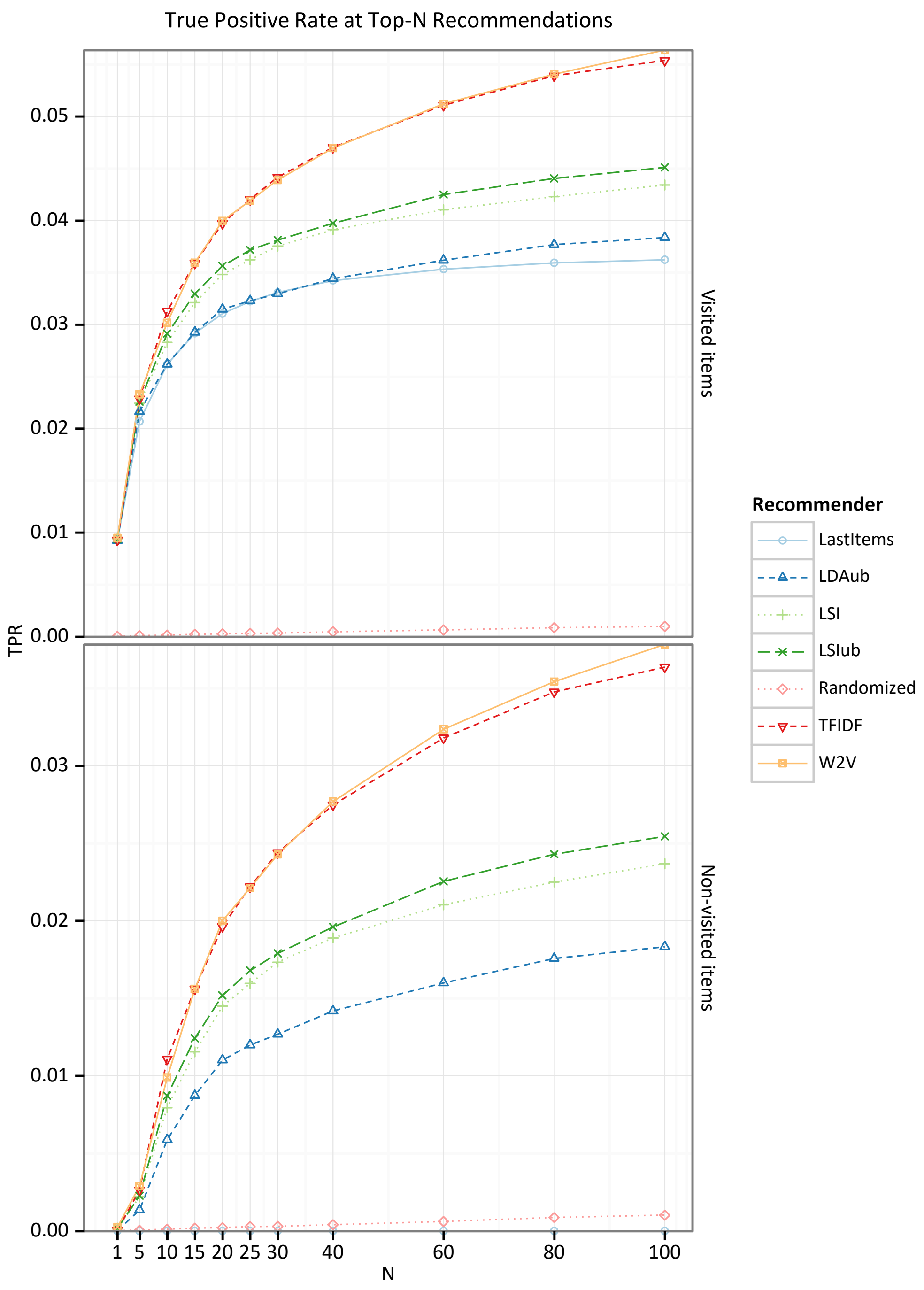


Figure 17: True-Positive-Rate for visited and non-visited items at different recommendation list sizes N.

The precision for visited and non-visited items at different recommendation list sizes is shown in Figure 18. For visited items, LastItems recommender is the best performer. Thus, the proportion of correctly predicted items and all items recommended by the recommender is the highest for LastItems. TFIDF and W2V perform similar to each other, being better than all others but LastItems. LSI and LSIub also perform similar to each other, being slightly worse than TFIDF and W2V but better than LDAub and Randomized. LDAub is the worst performer from the VSM recommenders. However, all the recommenders perform remarkably well in comparison to Randomized recommender.

For non-visited items, the results are similar to those of the TPR for non-visited items, where TFIDF and W2V are the top performers, followed by LSIub and LSI, then LDAub, and finally weak performers Randomized and LastItems. Since LastItems does not recommend any new items, the precision is zero. Once again, the recommendation of top five most recently experienced items is reflected in the non-visited items figure. For most of the recommenders, the precision is the highest when the recommendation list size is around 15.

Figure 18 also shows that precision decreases when the recommendation list size increases. This applies for all recommenders. This means that the more items are recommended, the less is the chance that the predicted items are correct. Moreover, it is possible to increase the recommendation list size even when all items to be predicted have already been predicted by the recommender, which decreases the precision. Overall, the precision for non-visited items is significantly lower for non-visited items than visited items.

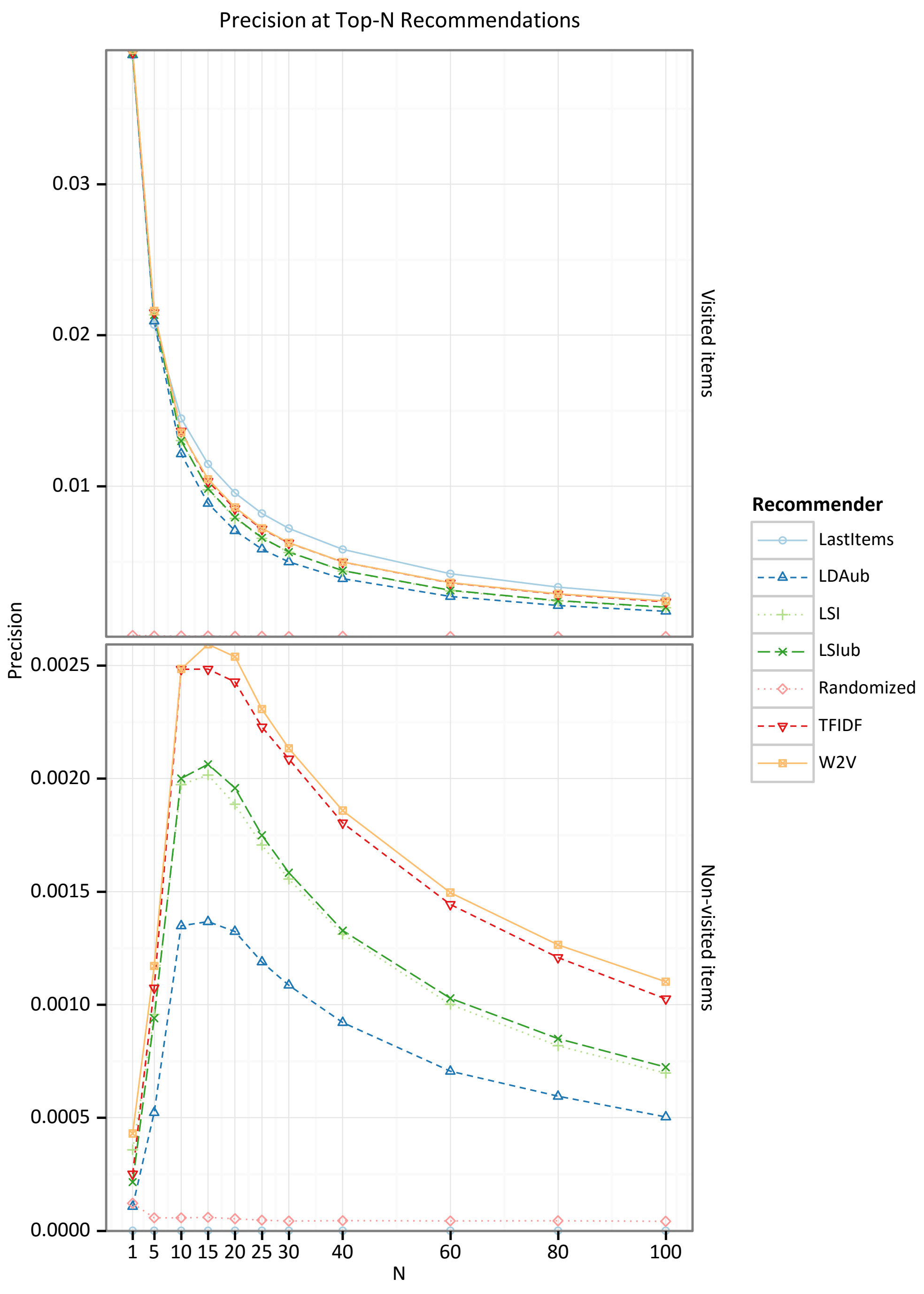


Figure 18: Precision for visited and non-visited items at different recommendation list sizes N.

The F1-Score for visited and non-visited items at different recommendation list sizes is shown in Figure 19. F1-Score indicates an overall utility of the recommendations, taking into account both precision and TPR.

For visited items, LastItems is the best performer, and all the VSM recommenders perform similar to each other, being slightly worse than LastItems. Randomized recommender is clearly the worst performer.

For non-visited items, the F1-Score for all recommenders but Randomized perform similar to each other. Randomized recommender is the worst performer. The F1-Score for LastItems is not defined, since it would require division by zero because of the non-visited items.

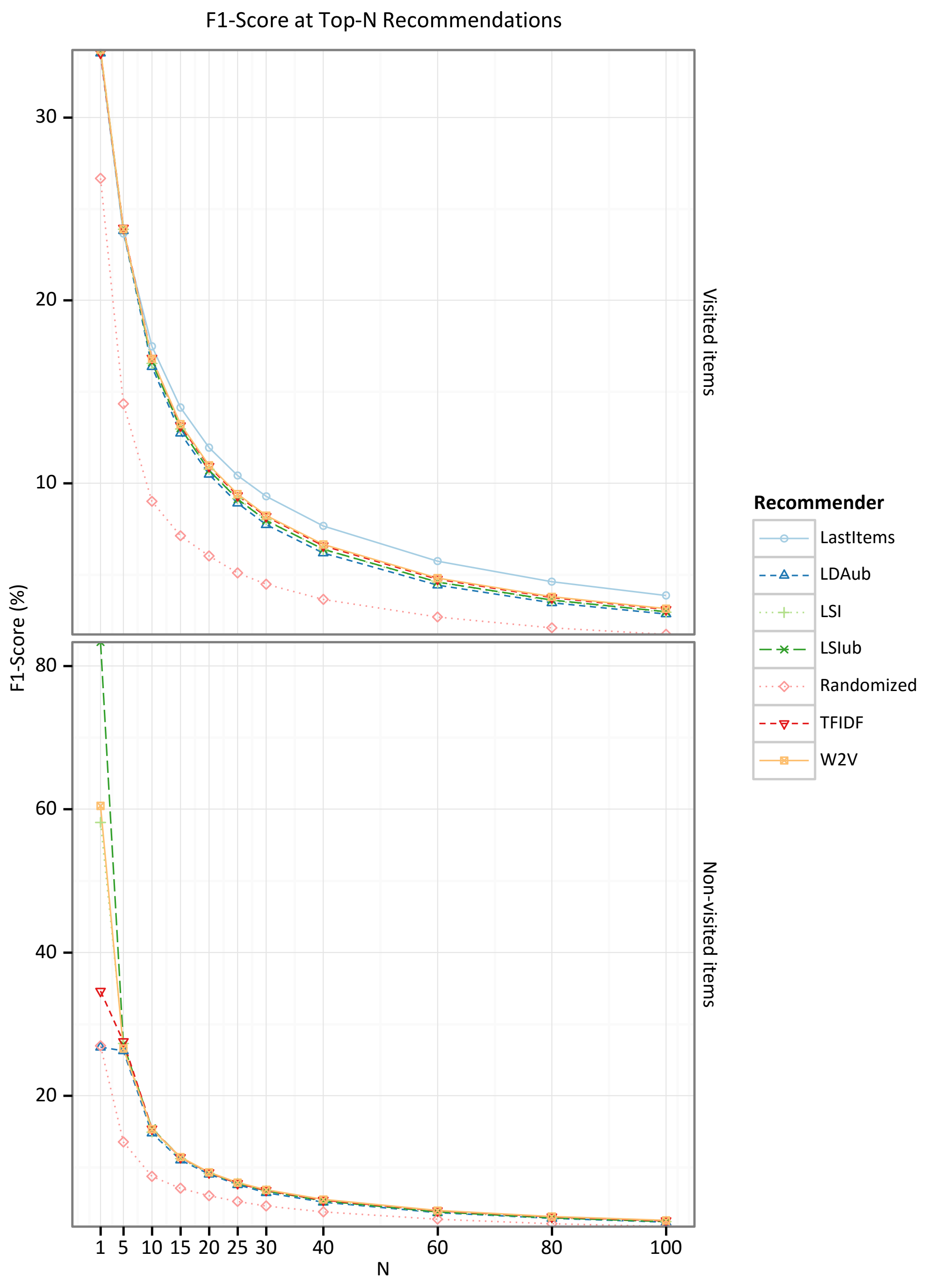


Figure 19: F1-Score for visited and non-visited items at different recommendation list sizes N.

The results for ranking metrics R20, R100 and NDCG for different recommenders are shown in Figure 20. These rankings are for visited items only. Both TFIDF and W2V are the top performers on all the ranking metrics, whereas the reference recommenders LastItems and Randomized are the worst performers together with LDAub. LSIub and LSI are placed in the middle performance-wise.

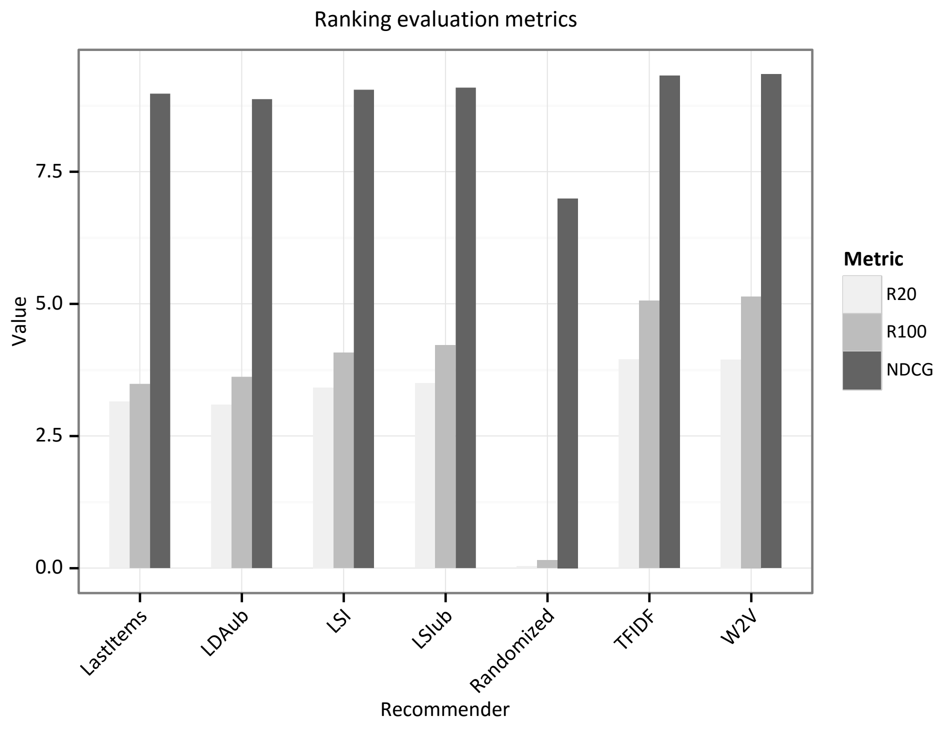


Figure 20: R20, R100 and NDCG evaluation metrics for each recommender.

The average time spent in making the recommendation is visualized in Figure 21, and the corresponding numerical values are presented in Table 9. From these it can be observed that all the recommenders have prediction time less than 40 milliseconds. The total time required for recommendation in the case of reference recommenders LastItems and Randomized is very low. The prediction and aggregation times of VSM recommenders are similar, apart from W2V that is significantly slower than others, and TFIDF that is slightly faster than others.

Table 9: Average time spent in recommendation.

Recommender	Aggregation time (ms)	Prediction time (ms)	Total time (ms)
LastItems	0,98	1,95	2,93
LDAub	71,06	11,47	82,53
LSI	67,68	12,86	80,54
LSIub	68,14	12,87	81,02
Randomized	4,73	7,09	11,82
TFIDF	48,71	10,49	59,19
W2V	70,58	36,17	106,76

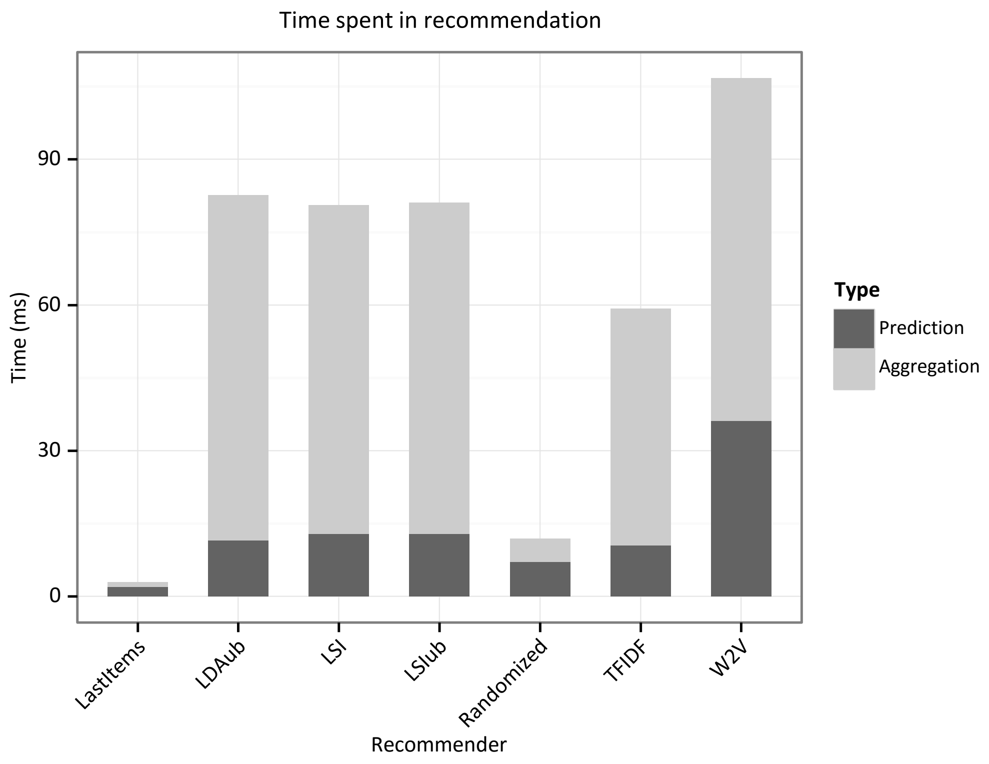


Figure 21: Average time spent in recommendation for each recommender.

6 Discussion and conclusions

The goal of this thesis was to make the front page of an online auction site more relevant to its users by building a recommender system that chooses the items to display. A general recommendation process was developed to serve as a basis for several recommenders that were built. Randomized and LastItems recommenders were based on simple heuristics and they were used as reference points for other recommenders. TFIDF, LSI, LSIub, LDAub and W2V recommenders were based on representing user profile and items in the respective vector space of each recommender. The recommenders were evaluated offline by simulating them in a real-life scenario for one week. The predictive accuracy was used as an evaluation metric for both visited and non-visited items, and the recommendation time was also considered.

The results show that the simple LastItems recommender performed surprisingly well for visited items. This shows that users tend to revisit the items they have already visited. Therefore, it would make sense to display at least a small number of previously visited items on the front page. Initial experiments with different recommenders already showed this kind of behavior from users, and it is also the reason for ranking previously visited items in the top five in the recommendation process.

Revisiting the same items could be for various reasons. For instance, users may want to keep up with the bids and comments of an interesting auction item. On the other hand, sellers may want to do the same. Simply recommending previously visited items should make the front page more relevant, and also more userfriendly by providing a shortcut to a potentially interesting item. With more extensive data available it would be possible to examine possible reasons for this kind of behavior in more detail.

On the other hand, it would be uninteresting for the users to only see previously visited on the front page. Therefore, it would make sense to also recommend unseen items for the users. Based on the results, the best performers for non-visited items were TFIDF and W2V. Interestingly, these recommenders are based on totally different approaches. TFIDF is more of a content-based approach that only takes into account exact words in the item titles, whereas W2V uses more collaborative-based approach by taking into account neighboring words in users' sessions in general. The accuracy of W2V could be enhanced by using more advanced heuristics for combining individual words, such as Paragraph2Vec. Moreover, increasing the dimensionality d_{W2V} could lead to better results, but it would also increase the required computation time. Because W2V has to aggregate individual words in to sentences, on average it takes almost double the time of TFIDF to compute a recommendation. Therefore, TFIDF is more scalable than W2V.

For the online auction site, W2V could be an interesting approach with many extension possibilities. Because the online auction is owned by a large media corporation, W2V would allow for using user data from other sources than the auction site as well. For example, if there were data about news articles that users have read or Facebook comments that users have posted, this information could be exploited by W2V since it tries to identify the contexts of words. Moreover, the average performing topic model recommenders LSI and LDA could also perform better when trained with more extensive data.

The more traditional TFIDF recommender was simple, efficient, and performed well, even with limited amount of data available, as in this thesis. However, the recommendations of TFIDF are based on exact word matches, which users may consider uninteresting in reality. On the other hand, TFIDF recommender performs search queries based on words in previous sessions of a user, and items similar to those previously visited are obtained as a result. Due to its simplicity and scalability, TFIDF recommender is a good candidate for further examination and online evaluation.

One of the main challenges in this thesis was the uniqueness and dynamic nature of items. In this thesis, the approach to tackle this problem was to offline calculate user profile and online calculate item similarities in the same vector space, with the help of cosine similarity and natural language processing based recommenders. Static item categories were used for fetching the candidate items to calculate similarities against. One approach that was not comprehensively considered in this thesis was clustering. Item categories were used as pre-defined static clusters, but by clustering items based on other types of item features, such as location and price, it would be possible to create more specific clusters and apply more traditional recommendation techniques that require more computation time. For example, when users add items for sale, these items could automatically be clustered into more detailed, pre-defined static clusters. Traditional recommendation techniques could then be applied to these static clusters instead of single items, and items could be chosen from these clusters based on some heuristics.

The recommendation process in this thesis was based on initial experiments with possible approaches in order to meet the desired time and scalability requirements. The process utilizes both offline and online calculation which allows for producing recommendations quickly. In fact, the total aggregation and prediction time is under 100 milliseconds for almost every recommender, which indicates that user profiles could be aggregated near real-time, and the information of the current session could also be utilized when computing the recommendations. Moreover, for most recommenders, more candidate items or user word queries could be used while still reaching the target prediction time.

On the other hand, the recommendation process only takes into account the most recent sessions of a user. For some users this could be sufficient, but for some users it could be beneficial to create a longer-term user profile that takes a larger portion of the user history into account. However, creating a long-term profile would be more useful with more extensive user data available, such as previous items bought and sold.

Because the recommenders used in this thesis were mostly content-based, recommendations can easily be made for users with only a few events in their user history. Moreover, since category similarities were calculated using a collaborative approach, new categories can be recommended to users with experience from only few categories. These choices aim to tackle the cold-start problem.

As mentioned, the data used in this thesis were limited. User browsing history was used as an input for the recommenders, and on the item side, only the item titles and categories were utilized. Given that users’ previous sales, bids, purchases, feedbacks and demographic information was available with extensive item information, recommenders with more predictive accuracy could be built. This extra information could be utilized, for example, in the item similarity calculation by adding the extra information as new features in the vector representation of the item. However, adding more features would increase the recommendation computation time. Another use case for additional information could be the creation of a more extensive user profile that could be pre-calculated offline. This profile information could be utilized when constructing the final rankings for a smaller number of items (as in the YouTube example in Section 2.3.1), for example by ranking those items higher that are located closer to the user. Information about users' previous purchases would also allow for obvious post-purchase recommendations, such as recommending an iPad cover after buying an iPad.

Appendix A – Table of evaluation results

Recommender	N	TPR	TPR (non-visited)	Precision	Precision (non-visited)	F1-Score	F1-Score (non-visited)
LastItems	1	0,0092	0,0000	0,0385	0,0000	0,3352	-
LastItems	5	0,0207	0,0000	0,0207	0,0000	0,2365	-
LastItems	10	0,0262	0,0000	0,0145	0,0000	0,1748	-
LastItems	15	0,0292	0,0000	0,0115	0,0000	0,1413	-
LastItems	20	0,0311	0,0000	0,0096	0,0000	0,1195	-
LastItems	25	0,0322	0,0000	0,0082	0,0000	0,1042	-
LastItems	30	0,0331	0,0000	0,0072	0,0000	0,0928	-
LastItems	40	0,0342	0,0000	0,0058	0,0000	0,0766	-
LastItems	60	0,0353	0,0000	0,0042	0,0000	0,0574	-
LastItems	80	0,0359	0,0000	0,0033	0,0000	0,0462	-
LastItems	100	0,0362	0,0000	0,0027	0,0000	0,0386	-
LDAub	1	0,0092	0,0000	0,0386	0,0001	0,3351	0,2679
LDAub	5	0,0217	0,0014	0,0209	0,0005	0,2380	0,2626
LDAub	10	0,0262	0,0059	0,0121	0,0013	0,1636	0,1478
LDAub	15	0,0293	0,0087	0,0089	0,0014	0,1273	0,1103
LDAub	20	0,0315	0,0110	0,0071	0,0013	0,1048	0,0906
LDAub	25	0,0323	0,0120	0,0058	0,0012	0,0890	0,0758
LDAub	30	0,0330	0,0127	0,0050	0,0011	0,0773	0,0646
LDAub	40	0,0344	0,0142	0,0039	0,0009	0,0618	0,0514
LDAub	60	0,0362	0,0160	0,0027	0,0007	0,0442	0,0367
LDAub	80	0,0377	0,0176	0,0021	0,0006	0,0346	0,0288
LDAub	100	0,0384	0,0183	0,0017	0,0005	0,0284	0,0235
LSI	1	0,0094	0,0002	0,0388	0,0004	0,3364	0,5816

LSI	5	0,0224	0,0022	0,0213	0,0010	0,2389	0,2731
LSI	10	0,0283	0,0079	0,0130	0,0020	0,1654	0,1494
LSI	15	0,0321	0,0115	0,0098	0,0020	0,1296	0,1125
LSI	20	0,0348	0,0145	0,0079	0,0019	0,1075	0,0915
LSI	25	0,0362	0,0160	0,0065	0,0017	0,0916	0,0779
LSI	30	0,0375	0,0173	0,0056	0,0016	0,0800	0,0674
LSI	40	0,0391	0,0189	0,0044	0,0013	0,0637	0,0530
LSI	60	0,0410	0,0210	0,0031	0,0010	0,0457	0,0379
LSI	80	0,0423	0,0225	0,0024	0,0008	0,0357	0,0295
LSI	100	0,0434	0,0237	0,0020	0,0007	0,0294	0,0245
LSIub	1	0,0094	0,0002	0,0387	0,0002	0,3366	0,8333
LSIub	5	0,0226	0,0023	0,0213	0,0009	0,2391	0,2718
LSIub	10	0,0291	0,0087	0,0130	0,0020	0,1661	0,1545
LSIub	15	0,0330	0,0124	0,0098	0,0021	0,1302	0,1142
LSIub	20	0,0356	0,0152	0,0079	0,0020	0,1070	0,0926
LSIub	25	0,0372	0,0168	0,0066	0,0017	0,0913	0,0779
LSIub	30	0,0381	0,0179	0,0056	0,0016	0,0796	0,0673
LSIub	40	0,0397	0,0196	0,0044	0,0013	0,0639	0,0533
LSIub	60	0,0425	0,0225	0,0031	0,0010	0,0459	0,0380
LSIub	80	0,0440	0,0243	0,0024	0,0008	0,0361	0,0297
LSIub	100	0,0451	0,0255	0,0020	0,0007	0,0297	0,0244
Randomized	1	0,0000	0,0000	0,0001	0,0001	0,2667	0,2698
Randomized	5	0,0001	0,0000	0,0001	0,0001	0,1434	0,1357
Randomized	10	0,0002	0,0001	0,0001	0,0001	0,0900	0,0876
Randomized	15	0,0002	0,0002	0,0001	0,0001	0,0712	0,0710
Randomized	20	0,0003	0,0002	0,0001	0,0001	0,0602	0,0607
Randomized	25	0,0003	0,0003	0,0001	0,0000	0,0509	0,0527
Randomized	30	0,0003	0,0003	0,0001	0,0000	0,0448	0,0463
Randomized	40	0,0005	0,0004	0,0001	0,0000	0,0364	0,0381
Randomized	60	0,0006	0,0006	0,0001	0,0000	0,0268	0,0279

Randomized	80	0,0009	0,0009	0,0001	0,0000	0,0209	0,0216
Randomized	100	0,0010	0,0010	0,0000	0,0000	0,0174	0,0178
TFIDF	1	0,0093	0,0001	0,0387	0,0003	0,3353	0,3460
TFIDF	5	0,0229	0,0026	0,0215	0,0011	0,2395	0,2756
TFIDF	10	0,0313	0,0111	0,0136	0,0025	0,1683	0,1540
TFIDF	15	0,0359	0,0156	0,0103	0,0025	0,1313	0,1140
TFIDF	20	0,0397	0,0196	0,0085	0,0024	0,1091	0,0930
TFIDF	25	0,0420	0,0222	0,0072	0,0022	0,0932	0,0778
TFIDF	30	0,0442	0,0244	0,0062	0,0021	0,0818	0,0680
TFIDF	40	0,0470	0,0275	0,0050	0,0018	0,0657	0,0546
TFIDF	60	0,0511	0,0318	0,0036	0,0014	0,0475	0,0396
TFIDF	80	0,0539	0,0348	0,0028	0,0012	0,0374	0,0311
TFIDF	100	0,0554	0,0364	0,0023	0,0010	0,0309	0,0257
W2V	1	0,0095	0,0003	0,0389	0,0004	0,3370	0,6045
W2V	5	0,0233	0,0029	0,0216	0,0012	0,2389	0,2660
W2V	10	0,0302	0,0099	0,0136	0,0025	0,1680	0,1533
W2V	15	0,0359	0,0156	0,0105	0,0026	0,1323	0,1144
W2V	20	0,0400	0,0200	0,0086	0,0025	0,1097	0,0931
W2V	25	0,0419	0,0221	0,0072	0,0023	0,0940	0,0789
W2V	30	0,0439	0,0243	0,0063	0,0021	0,0822	0,0687
W2V	40	0,0470	0,0277	0,0050	0,0019	0,0664	0,0553
W2V	60	0,0512	0,0324	0,0036	0,0015	0,0480	0,0398
W2V	80	0,0541	0,0354	0,0029	0,0013	0,0379	0,0314
W2V	100	0,0564	0,0378	0,0024	0,0011	0,0313	0,0261

Bibliography

Amatriain, X. & Basilico, J., 2012. *The Netflix Tech Blog.* [Online] Available at: http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html [Accessed 12 6 2015].

Barbieri, N., Manco, G. & Ritacco, E., 2014. Probabilistic Approaches to Recommendations. In: J. Han, et al. eds. *Synthesis Lectures on Data Mining and Knowledge Discovery.* San Rafael, California: Morgan & Claypool Publishers.

Becherer, R. C. & Halstead, D., 2004. Characteristics and Internet Marketing Strategies of Online Auction Sellers. *International Journal of Internet Marketing and Advertising,* 1(1), pp. 24-37.

Beel, J., Langer, S., Nürnberger, A. & Genzmehr, M., 2013. *Persistence in Recommender Systems: Giving the Same Recommendations to the Same Users Multiple Times.* Valletta, Malta, Springer, pp. 390-394.

Bhuiyan, T., 2013. *Trust for Intelligent Recommendation.* New York: Springer.

Bishop, C., 2006. *Pattern Recognition and Machine Learning.* 1 ed. New York: Springer-Verlag.

Blei, D. M., Ng, A. Y. & Jordan, M. I., 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research,* Volume 3, pp. 993-1022.

Bobadilla, J., Ortega, F., Hernando, A. & Gutiérrez, A., 2013. Recommender Systems Survey. *Knowledge Based Systems,* July, Volume 46, pp. 109-132.

Breese, J. S., Heckerman, D. & Kadie, C., 1998. *Empricial Analysis of Predictive Algorithms for Collaborative Filtering.* San Francisco, Morgan Kaufmann Publishers Inc., pp. 43-52.

Burke, R., 2007. Hybrid Web Recommender Systems. In: P. Brusilovsky, A. Kobsa & W. Nejdl, eds. *The Adaptive Web.* Berlin: Springer Berlin Heidelberg, pp. 377-408.

ChoiceStream, 2008. *ChoiceStream Personalization Survey 2008.* [Online] Available at: http://choicestream.com/2013_Staging/wp-

content/uploads/2013/10/ChoiceStream_2008_Personalization_survey.pdf
[Accessed 16 6 2016].

Davidson, J. et al., 2010. *The YouTube Video Recommendation System.* New York, ACM New York, pp. 293-296.

Dean, A. & Voss, D., 1999. *Design and Analysis of Experiments.* 1 ed. New York: Springer-Verlag New York.

Deerwester, S. et al., 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science,* 41(6), pp. 391-407.

Drake, J., 2007. *Important Auction Characteristics in E-Marketplace Decisions: An Exploratory Look At Auction Selection And Product Valuation.* Florida, s.n.

Drake, J. R., Hall, D. J., Cegielski, C. & Byrd, T. A., 2015. An Exploratory Look at Early Online Auction Decisions: Extending Signal Theory. *Journal of Theoretical and Applied Electronic Commerce Research,* 10(1), pp. 35-48.

eBay, 2015. *Electronics, Cars, Fashion, Collectibles, Coupons and More Online Shopping | eBay.* [Online]
Available at: http://www.ebay.com
[Accessed 16 6 2016].

Goldberg, D., Nichols, D., Oki, B. M. & Terry, D., 1992. Using collaborative filtering to weave an information tapestry. *Communications of the ACM - Special issue on information filtering,* 35(12), pp. 61-70.

Griffiths, T. L. & Steyvers, M., 2004. Finding Scientific Topics. *Proceedings of the National Academy of Sciences of the United States of America,* Volume 101, pp. 5228-5235.

Guo, G., 2012. *Systems, Resolving Data Sparsity and Cold Start in Recommender.* Heidelberg, Springer-Verlag Berlin, pp. 361-364.

Gündüz-Ögüdücü, Ş., 2010. Web Page Recommendation Models: Theory and Algorithms (Part of Synthesis Lectures on Data Management). In: M. Z. Özsoyoğlu & T. M. Özsu, eds. *Synthesis Lectures on Data Management.* 1 ed. s.l.:Morgan & Claypool Publishers.

Herlocker, J. L., Konstan, J. A., Terveen, L. G. & Riedl, J. T., 2004. Evaluating Collaborative Filtering Recommendation Systems. *ACM Transactions on Information Systems,* 22(1), pp. 5-53.

Huang, Z., Chung, W., Ong, T.-H. & Chen, H., 2002. *A Graph-based Recommender System for Digital Library.* New York, ACM New York, pp. 65-73.

Häubl, G. & Trifts, V., 2000. Consumer Decision Making in Online Shopping Environments: The Effects of Interactive Decision Aids. *Marketing Science,* 19(1), pp. 4-21.

Jank, W. & Shmueli, G., 2010. *Modeling Online Auctions.* Hoboken, N.J.: John WIley & Sons.

Järvelin, K. & Kekäläinen, J., 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems (TOIS),* October, 20(4), pp. 422-446.

Katukuri, J., Mukherjee, R. & Konik, T., 2013. *Large-Scale Recommendations in a Dynamic Marketplace.* Hong Kong, s.n.

Krishna, V., 2002. *Auction Theory.* 2 ed. San Diego: Academic Press.

Kusner, M. J., Sun, Y., Kolkin, N. I. & Weinberger, K. Q., 2015. *From Word Embeddings To Document Distances.* Lille, France, s.n.

Lam, X. N., Vu, T., Le, T. D. & Duong, A. D., 2008. *Addressing Cold-start Problem in Recommendation Systems.* New York, ACM New York, pp. 208-211.

Landauer, T. K., Foltz, P. W. & Laham, D., 1998. An Introduction to Latent Semantic Analysis. *Discourse Processes,* Volume 25, pp. 259-284.

Linden, G., Smith, B. & York, J., 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing,* 7(1), pp. 76-80.

Li, X., Xia, G., You, W. & Xhang, Z., 2007. *Recommendation of Online auction Items Focusing Collaborative Filtering.* Shanghai, IEEE, pp. 6191-6194.

Lops, P., de Gemmis, M. & Semeraro, G., 2011. Content-based Recommender Systems: State of the Art and Trends. In: F. Ricci, L. Rokach, B. Shapira & P. B. Kantor, eds. *Recommender Systems Handbook.* s.l.:Springer US, pp. 73-105.

Mikolov, T., Chen, K., Corrado, G. & Dean, J., 2013. *Efficient Estimation of Word Representations in Vector Space.* s.l., s.n.

Mikolov, T. & Le, Q., 2014. *Distributed Representations of Sentences and Documents.* Beijing, China, s.n., pp. 1188-1196.

Narayanan, A. & Shmatikov, V., 2008. *Robust De-anonymization of Large Sparse Datasets.* Washington DC, IEEE Computer Society, pp. 111-125.

Park, D. H., Kim, H. K., Choi, I. Y. & Kim, J. K., 2012. A literature review and classification of recommender systems research. *Expert Systems with Applications,* 39(11), pp. 10059-10072.

Parsons, J., Ralph, P. & Gallagher, K., 2004. *Using Viewing Time to Infer User Preference in Recommender Systems.* s.l.:Proceedings of the AAAI Workshop in Semantic Web Personalization (California, 2004).

Pazzani, M. J. & Billsus, D., 2007. Content-based Recommendation Systems. In: P. Brusilovsky, A. Kobsa & W. Nejdl, eds. *The Adaptive Web.* s.l.:Springer Berlin Heidelberg, pp. 325-341.

Pinckney, T., 2013a. *NYC* 2013 - "Graph-based Recommendation Systems at eBay" by Thomas Pinckney.* [Online]
Available at: https://www.youtube.com/watch?t=2400&v=Tg3dP2fZGSM
[Accessed 6 7 2015].

Pinckney, T., 2013b. *Planet Cassandra: Graph Based Recommendation Systems at eBay.* [Online]
Available at: http://www.slideshare.net/planetcassandra/e-bay-nyc
[Accessed 6 7 2015].

Řehůřek, R., 2016. *Gensim: Topic Modelling For Humans.* [Online]
Available at: https://radimrehurek.com/gensim/
[Accessed 16 6 2016].

Resnick, P. & Varian, H. R., 1997. Recommender systems. *Communications of the ACM,* 40(3), pp. 56-58.

Resnick, P., Zeckhauser, R., Swanson, J. & Lockwood, K., 2006. The value of reputation on eBay: A controlled experiment. *Experimental Economics,* 9(2), pp. 79-101.

Ricci, F., Rokach, L., Shapira, B. & Kantor, P. B., 2011. Introduction to Recommender Systems Handbook. In: F. Ricci, L. Rokach & S. Bracha, eds. *Recommender Systems Handbook.* 1 ed. s.l.:Springer US, pp. 1-35.

Rijsbergen, C. J. V., 1979. *Information Retrieval.* 2 ed. MA, USA: Butterworth-Heinemann Newton.

Roth, A. E. & Ockenfels, A., 2002. Last Minute Bidding and the Rules for Ending Second-Price Auctions. *American Economic Review,* 4(92), pp. 1093-1103.

Samuelson, W., 2014. Auctions: Advances in Theory and Practice. In: K. Chatterjee & W. Samuelson, eds. *Game Theory and Business Applications.* 2nd edition ed. New York: Springer US, pp. 328-366.

Sanoma, 2015. *Huuto.net-verkkohuutokauppa.* [Online]
Available at: http://www.huuto.net
[Accessed 16 7 2015].

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J., 2001. *Item-based Collaborative Filtering Recommendation Algorithms.* New York, ACM New York, pp. 285-295.

Schafer, J. B., Konstan, J. A. & Riedl, J., 2001. E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery,* 5(1-2), pp. 115-153.

Schafer, J. B., Konstan, J. & Riedl, J., 1999. *Recommender Systems in E-Commerce.* New York, ACM New York, pp. 158-166.

Schein, A. I., Popescul, A., Ungar, L. H. & Pennock, D. M., 2002. *Methods and Metrics for Cold-start Recommendations.* New York, ACM New York, pp. 253-260.

Schwartz, B., 2004. *The Paradox of Choice.* New York: Harper Collins.

Shani, G. & Gunawardana, A., 2011. Evaluating Recommendation Systems. In: F. Ricci, L. Rokach & S. Bracha, eds. *Recommender Systems Handbook.* 1 ed. s.l.:Springer US, pp. 257-297.

Uchyigit, G. & Ma, M. Y., 2008. *Personalization Techniques and Recommender Systems.* Hackensack, NJ: World Scientific.

Wang, J.-C. & Chiu, C.-C., 2008. Recommending Trusted Online Auction Sellers Using Social Network Analysis. *Expert Systems with Applications,* 34(3), pp. 1666-1679.

Yoo, K.-H., Gretzel, U. & Zanker, M., 2013. *Persuasive Recommender Systems.* s.l.:Springer-Verlag New York.