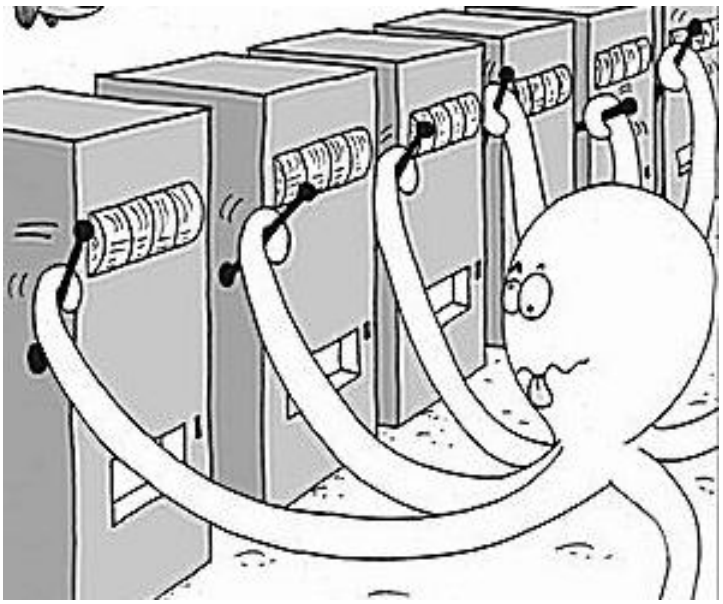
CHALMERS



Recommender Systems; Contextual Multi-Armed Bandit Algorithms for the purpose of targeted advertisement within e-commerce

Master of Science Thesis in Computer Science: Algorithms, Languages and Logic

Fredrik Ek
Robert Stigsson

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2015
Master's Thesis 2015

Recommender Systems
Contextual Multi-Armed Bandit Algorithms for the purpose of targeted advertisement within e-commerce

Fredrik Ek,
Robert Stigsson

Examiner: Devdatt Dubhashi

Supervisor: Christos Dimitrakakis

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: A squid playing on multiple one-armed bandits at once, modelling the idea of the Multi-Armed Bandit Algorithm.

This thesis has been prepared using LaTeX.

Department of Computer Science and Engineering
Gothenburg, Sweden June 2015

Abstract

The topic of Recommender Systems is and have been a hot topic the last century as the market for e-commerce keeps extending. Basic techniques for recommending popular items are used on more or less all e-commerce platforms. Many e-commerce based platforms use simple techniques such as ”people who bought this also bought that”, while others have very complex recommender systems for customised recommendations depending on users profiles. Regardless of what techniques that are being used, companies want to make their customers happy as well as increasing their own profit. Because of this there is a constant demand for smart systems using up-to-date algorithms and techniques to achieve relevant advertisements. This thesis focuses on evaluating the performance of Contextual Multi-Armed Bandit Algorithms in a, for the specific algorithm, not yet fully explored use-area of recommender systems, namely the area of garment-based e-commerce.

The evaluation consists in measuring the performance mostly in terms of successful recommendations, while discussing satisfaction-level of customers. In addition to this we decided to experiment with different privacy-preserving techniques to see how it affects the performed recommendations. This kind of evaluation is, to our knowledge, absent in current literature, which is why we decided to pursue the idea.

The evaluation is carried out through use of self-implemented algorithms. Using the frameworks for machine-learning and implementing recommender systems, Apache Mahout and LensKit, the algorithms used in this thesis are implemented by ourselves in java.

The implemented algorithms turned out to be better than what was expected initially, managing to predict purchase-behavior of some users with a probability of over 21%

Through observing the results of the implemented application we made it possible to identify new possible use-areas for Multi-Armed Bandit Algorithms within the topic of Recommender Systems.

Acknowledgements

First, we would like to thank our supervisor at Consid AB in Gothenburg, Petter Bergström, for assisting us in coming up with the Master's thesis proposal and giving us the opportunity of performing it at Consid AB. But also for getting us in touch with Junkyard Trading AB who could provide us with the dataset, without which we would not be able to go through with the actual thesis idea.

Next, we would like to thank Junkyard Trading AB for actually allowing us to use the data from their live e-commerce platform.

Finally, we would like to thank our supervisor at Chalmers University of Technology, Christos Dimitrakakis, for helping us in composing and finalising the proposal and for continuous feedback throughout the project. Without his expertise we would not have managed as well. We would also like to thank our examiner at Chalmers University of Technology, Devdatt Dubhashi, for his help and feedback.

Fredrik Ek and Robert Stigsson, Gothenburg 2015

Contents

1

Introduction

This thesis is performed at Chalmers University of Technology in collaboration with and working in the facilities of Consid AB. The purpose of this thesis is to provide a thorough analysis and evaluation of the algorithmic approach of Contextual Multi-Armed Bandit Algorithms for the use within e-commerce, while highlighting the recurring topic of privacy within Computer Science. The goal is to perform several types of evaluations through implementation of different algorithms. Implementations in this project make use of the frameworks for machine-learning and developing and evaluating recommender systems, LensKit[1] and Mahout[2], which will be reviewed further under 5.7. Specifics regarding the implemented evaluation methods can be found under 5.8 and 5.9

1.1 Background

Recommender Systems is a hot topic within Computer Science, as algorithms for the purpose of targeted advertisement[3] get improved each year. Hence it is vital for companies to keep up with their competitors. Personal integrity, or online privacy, is also a frequently discussed topic within computer science, as well as, in society and on a global level. How much and what kind of data is ethically acceptable to observe in order to produce relevant advertisements? Is it acceptable to use all data that could be retrieved from a user such as GPS-based movement, cookies, chat-history, previous purchases or other arbitrary personal data a company might have access to?

This thesis focuses on the evaluation and analysis of the performance of Contextual Multi-Armed Bandit Algorithms for the purpose of targeted advertisement within e-commerce. Comparisons are made with more standard algorithms, being used in many live recommender systems for e-commerce as of today. Whereas the algorithmic approach of Contextual Multi-Armed Bandit algorithms have use-areas within recom-

mender systems[4], very little research has been made for use within e-commerce. Performance will also be analysed and evaluated taking into consideration how much personal data are observed − How much and what kind of data can be observed and manipulated for it to still be considered socially and ethically acceptable? That is while still respecting the users' privacy. There exist research on the subject, addressing the issue of online privacy[5].

The ideas of the Multi-Armed Bandit problem originates from gambling and slot-machines (one-armed bandits), expressing a statistical decision model of an agent trying to choose in which order to play on a set of different machines[6, 7]. More information of how The Multi-Armed Bandit Algorithm works can be found under 3.2.

We have established a connection with the IT-company Consid AB. Many of their clients are looking for e-commerce based platforms, which they develop using well-established content management systems (CMS). Consid are thereby interested in a proof-of-concept application using data from one of their developed e-commerce platforms. If this proof-of-concept turns out well, there is a chance that it could be sold as a product. Consid have five years of data stored from a live e-commerce system they developed for the company Junkyard Trading AB. Data that we may use in any way we like.

1.2 Methodology

The development of software for this project is done in an agile fashion using sprints where each sprint consists of a planning and an evaluation phase. The sprints include developing and designing software for the recommender systems, as well as, testing and evaluating the considered algorithms.

The project has been carried out working in facilities of the IT-consultant company Consid AB in Gothenburg, where participants of the project have been working together in the same office.

Meetings with the supervisor from Chalmers have been scheduled and taken place every Friday, where follow-up on progression has been discussed.

Early steps of the thesis include a major research-based iteration, where the more important parts of upcoming implementation steps get thoroughly planned, before proceeding to the actual implementation. These important parts for instance consist in building the different models, used by the algorithms, as well as finding suitable evaluation methods. General knowledge about the used algorithms have also been gathered during these first iterations.

Research have furthermore composed big parts of the project as a whole, as each sprint required extensive research and planning.

The used frameworks have built-in support for the use of several common algorithms within the area of recommender systems such as: Item-based Collaborative Filtering, User-based Collaborative Filtering, Matrix factorization and Slope-One. However, no framework contains anything concerning Multi-Armed Bandit Algorithms. The frameworks are also supposed to contain built-in support for several evaluation methods.

Using these kind of frameworks also mean that time will not be wasted implementing data structures, interfaces and other handy functionality only for the purpose of a good visual experience when observing results etc. The focus could thereby rather be put on the algorithmic and evaluation aspects. Another advantage is that the used frameworks come with a well-documented java API, which is also one of the major reasons why they were chosen.

1.3 Related Work

In this section we briefly present some of the studied literature related to Multi-Armed Bandits, recommender systems, collaborative filtering, content-based filtering, demographic filtering, machine-learning, privacy and personalization.

The concept of Multi-Armed Bandits has been a well-studied area ever since it was first invented in 1952 [8] and several different approaches have been tried throughout the years [9, 10, 11, 12, 13]. Neither is the concept of recommender systems something new. Thorough research regarding recommender systems for e-commerce has been made throughout the years. As can be seen in [3], it was already a hot topic in the 1990s, whereas Markov Decision Processes, were already used for advertising by Howard back in 1960 [14]. As of late the use of *Contextual Multi-Armed Bandit algorithms* can be seen much in news-recommendation systems[15]. Contextual Multi-Armed Bandit Algorithms are typically implemented using methods of *content-based filtering*, which is another well-tested approach[16, 17] and together with *Collaborative Filtering* [18] and *Demographic Filtering* [19] compose the group of standard approaches[20] when implementing recommender systems.

One good example of a company making use of collaborative filtering based methods for their e-commerce platforms is Amazon.com [21]. Something else that is to be considered related work is the Netflix prize challenge solution [22], which has set a permanent mark in the area of recommender systems. The Multi-Armed Bandit Algorithm is, in contrary to collaborative filtering based algorithms, more likely to be implemented using approaches of content-based or perhaps demographic filtering. Content-Based Filtering algorithms make use of learning- and decision-policies which is another well-studied area with ϵ-*greedy* [23], *Upper Confidence Bounds*[23, 24] and the more recent *Thompson Sampling* [25, 26] as well-tested methods. Other than these, several well-established methods of machine-learning such as Bayesian techniques from [7] and [27] is to be con-

sidered and some even applied and deployed.

Another subject of this thesis is that of online privacy and moreover *privacy-preserving systems*, which is highly relevant when working with implementing recommender systems as all developers will be dealing with more or less sensitive private user-data. There have been research on the matter [5, 19, 28], most of which goes outside the scope of this thesis as focus are to be put on performance before privacy. See 1.4 for further information regarding the scope.

The biggest and most important part of this thesis is the evaluation part. It is also the hardest part, as different algorithmic approaches, techniques and methods depends on different things to be good, not least the features included in the observed datasets. There are however much recent and local work when it comes to the subject of recommender systems evaluation[29, 30, 31, 32].

1.4 Scope

The area of recommender systems is huge and used in many different environments, which is the reason why it is necessary to narrow this project down and specify a clear setting of what one want to achieve.

Algorithm design tends to vary quite a bit depending on the setting, and this thesis focuses on MABs, Multi-Armed Bandit Algorithms, in an area where little research has been made previously, the area of recommender systems intended for use within e-commerce. As Junkyard's primary business consists in e-commerce for clothing and accessories it is also relevant to limit this project to e-commerce for exactly this. Further, predictions for recommending products will be made keeping user experience in mind rather than making extra profit. That is, expensive products will not be recommended in favour of user satisfaction.

So far MABs have been actively pointed out to compose the core of this thesis. There are several algorithmic approaches within the area of recommender systems and moreover this thesis focus on a performance evaluation of a system using a MAB-based design setting. Other algorithmic approaches have however been used for comparison. The main, significantly different, approach for evaluation consists in an implementation of a Collaborative Filtering-based algorithm described under 5.8. Aside from this algorithm, several simple statistical models are also used for evaluation. More information regarding these can be found under 5.9.

This thesis does not aim to implement a complex set of algorithms with the purpose of beating the performance of a full-scale commercial recommender system such as ones used by Netflix, Amazon, Facebook etc. Thus the question is not whether a recommender system based on Contextual Multi-Armed Bandit algorithms with all it's belonging tech-

niques can beat the performance of the most accurate algorithm from the Netflix prize competition with it's ensemble of 107[22] different algorithmic approaches, blended into one complex implementation. But rather, would the algorithmic approach of contextual Multi-Armed Bandit algorithms perform well enough to belong among these different algorithmic approaches?
There are several major differences between the implemented recommender system for the Netflix prize and of what this thesis will focus on. More specific, the question for this thesis would be: When implementing a full-scale recommender system for use within e-commerce, is it worth considering Contextual Multi-Armed Bandit algorithms as one of the algorithmic approaches contributing towards a final solution?

Ethical and social aspects of privacy will be considered, but as the goal is not to implement a privacy-preserving recommender system, the discussion will be limited to analysing how performance changes taking a more privacy-preserving approach.

The scope of the dataset intended to be used in this project consists in the complete Junkyard database, used live in their e-commerce system from 2009 to 2014. We decided to limit the project to using only this, as it contains enough sufficient data. The Junkyard dataset is very contextual. There is a huge amount of data about each user and all of their purchases. The Junkyard dataset however has no data regarding click-sessions from ads or products.
Moreover, the Junkyard dataset initially contains information about several million users where the average number of orders made by each user is 3.52 and the average number of items purchased with each order being 2.24.

2

Recommender systems

Recommender Systems is and remains a hot topic within computer science in 2015. Huge conferences are hosted in different parts of the world annually, featuring many big companies. The RecSys conference [33] is a good example of this. Recommender Systems has a wide use-area within all kinds of advertisement on the web when specifically targeted ads are needed. These ads can be seen everywhere from websites and apps to emails, text-messages etc. In the list of big actors on the stage of recommender systems, hardly surprisingly, one find companies such as Facebook, Google etc. But Recommender Systems are also used for recommending music in for instance Spotify and movies/TV-shows in Netflix. There are a variety of different algorithms for each use-area. The setting of which this thesis will focus consists in algorithms for recommender systems within garment-based e-commerce. Here follows some of the more common general techniques used in recommender systems.

2.1 Collaborative filtering

The basic idea of collaborative filtering is to find information or patterns using collaborative techniques among multiple data sources. Data sources typically consist of users and items such as movies, songs etc. Within e-commerce, and this project specifically, those data sources mainly consist of users, and products that can be bought by these users. Finding these patterns is accomplished through collecting and analysing huge amounts of data on users' behaviours and activities as well as items. Within e-commerce those activities and behaviours include but are not limited to purchases and click-able advertisements. Collaborative filtering[18] is one of the most common general methods for recommender systems being applicable within several big use-areas. Collaborative Filtering is frequently used in social networks such as Facebook, LinkedIn, mySpace, Twitter etc to effectively recommend new friends, pages, groups as well as who to follow or what to like. But also applications such as Youtube, Reddit, Netflix etc make use of

collaborative filtering. Collaborative filtering is viable in all applications where one can observe connections between a user and their registered friends or followers. But it is also widely used within e-commerce, where Amazon is a good example who popularised algorithms for *item-to-item based collaborative filtering* [21].

Most Collaborative Filtering techniques can be expressed by the two general steps, Similarity Computation and Prediction Generation, described under 2.2.

2.1.1 User-based Collaborative Filtering

The main idea of recommender systems built on user-based collaborative filtering consists in computing the similarity between users' $\{u, j\}$ profiles, s_{uj}. That is, computing a prediction for the probability of the user u liking a specific item i, consists in computing a rating based on all ratings made by users with similar profiles. All similar profiles contribute to this prediction depending on the similarity factor, s_{uj} [34, 35, 36].

Typically this can be reduced to the two general steps.

1. While observing a user's actions in a system, look for similar users with equal behaviour- and activity-patterns.
2. Use the observations from step 1 to compute a prediction for the specified user.

See 2.2 for more information regarding Similarity Computation and Prediction Generation techniques.

2.1.2 Item-based Collaborative Filtering

The concept of item- based collaborative filtering applies the same idea as its User-based counterpart, but the similarity is computed between items instead of users, and is usually described as "Users who bought this also bought that"

More generally, taking an item-based approach means looking into the set, I, of items a specific user, u, has rated, using their context to compute the similarity of other items, $\{i_0, i_1, \ldots, i_n\}$, not in I. Their corresponding similarities are computed at the same time as $\{s_{i0}, s_{i1}, \ldots, s_{in}\}$. When these similarities and their corresponding items have been found, a prediction can be computed. [36, 37, 38].

This can typically be split into three steps:

1. Construct a User-Item matrix, $M[u][i]$, giving each index of the matrix a rating, r_{ui}, on an item i performed by a user u.
2. Compute the similarity, $s_{i1,i2}$, of two items i_1 and i_2 by looking at co-rated pairs from different users.

3. Generate predictions based on some prediction method, described under 2.2.

Item- based collaborative filtering is a very common approach, often used together with algorithms for *matrix factorization.*

2.1.3 Matrix Factorization

Matrix factorization [39] is an algebraic operation consisting in factorising a matrix $\mathbf{M}$, meaning finding the matrices $\mathbf{M_1}, \mathbf{M_2}, \ldots, \mathbf{M_3}$ such that when they are multiplied, the resulting matrix is $\mathbf{M}$.

In collaborative filtering based recommender systems this can be used as a rather simple and very intuitive algorithm for discovering latent features. By constructing a User-Item matrix M, where each index contains a rating r on an item i performed by a user u:

$$\mathbf{M_{u,i}} = \begin{pmatrix} r_{u1,i1} & r_{u1,i2} & \cdots & r_{u1,in} \\ r_{u2,i1} & r_{u2,i2} & \cdots & r_{u2,in} \\ \vdots & \vdots & \ddots & \vdots \\ r_{um,i1} & r_{um,i2} & \cdots & r_{um,in} \end{pmatrix} \tag{2.1}$$

For example, by limiting the matrix to 4×4 and adding some fictitious values for ratings where $-$ symbolises yet unrated items, a matrix to be factorised might look something like:

$$\mathbf{M_{u,i}} = \begin{pmatrix} 5 & 3 & - & 1 \\ 1 & - & - & 1 \\ 1 & - & - & 2 \\ 3 & 1 & 4 & 4 \end{pmatrix} \tag{2.2}$$

Matrix factorisation can be seen as the task of predicting the missing ratings meaning filling in the blanks $(-)$ in matrix 2.2. Latent features of items liked by different users might be different types of cloths or accessories which might be a shared interest between some users. So for the matrix, 2.2 above, an assumption regarding how many latent features one wishes to find must be made. Let us assume we want to find k latent features and we have the sets U and I containing all users and all items respectively. This means finding two matrices, $\mathbf{N}(a\ |U| \times k\ matrix)$ and $\mathbf{O}(an\ |I| \times k\ matrix)$ which when multiplied approximating $\mathbf{M}$:

$$\mathbf{M} \approx \mathbf{N} \times \mathbf{O} = \hat{\mathbf{M}} \tag{2.3}$$

Further, using similarity and prediction computing techniques as described under 2.2, the gaps can be filled in and thereby provide really good recommendations [39].

2.2 Similarity Computation and Prediction Generation

Similarity and prediction computations compose vital parts of collaborative filtering-based recommender systems. Something that can be seen in 2.1.1, 2.1.2 and 2.1.3.

The similarity computation is performed before the prediction generation and the basic idea is to first isolate users who have rated two different items, and secondly to apply some similarity computation technique between these two items to determine their similarity. There are several ways of computing the similarity but some common methods are Cosine-based Similarity, Correlation-based Similarity and Adjusted Cosine-based Similarity, Jaccard Similarity, Jaro-Winkler Similarity, Sörensen Similarity.

When the similarity computation is completed, it is time for the most important part in a collaborative filtering based recommender system; generating the output in terms of prediction. As for computing the similarity there are a number of techniques to choose from when generating predictions. Some of the more common methods are using regression or the weighted average/weighted sum.

2.3 Content-Based filtering

Another well known method when implementing recommender systems is content-based filtering[16, 17]. Content-based filtering is commonly used for movie recommendations or within other environments where for instance keywords are used to describe the items of the system. Among the actors of recommendation systems using content-based filtering one can find *The Internet Movie Database* and other popular websites for movies. The methods of Content-based filtering commonly make use of the correlation between item features and preferences from a user's profile, as opposed to the collaborative filtering approach that selects items based on the correlation between users with similar profiles.

For the above to work, the system needs to deploy some learning technique [16, 17] such as Bayesian networks, clustering, decision trees, neural networks, reinforcement learning, Nearest Neighbour etc. The system uses these techniques when observing historical data of users to learn their preferences. The intention is that, after sufficient amounts of data have been observed, the system should be able to predict future behaviour of a specific user.

2.3.1 Choosing a learning model

A key component when implementing a recommender system using content-based filtering is choosing a method for how the algorithm should be trained [16, 17], using available data from the system. Using this data to create a model of a user's preferences and history in the system can be seen as a kind of classification problem. There are several significantly different methods suitable for solving this, some of which are mentioned in

2.3. Which one to choose depend on the setting for which it is going to be used. For instance, Bayesian networks are commonly practical in a setting where knowledge about users change slowly, relative to time needed to construct the model. Clustering techniques on the other hand has a tendency to generate less-personal predictions than other methods[40]. Although, due to the nature of clustering, once the clustering process is complete, the new groups of data to be analyzed are significantly smaller than before and performance, in terms of computation time, are thereby likely to be good. Decision trees have shown a tendency of basing classifications on as few instances as possible. Something that has lead to worse performance in the sense of accuracy[41]. Although with a smaller number of structured attributes, the performance and simplicity of decision trees are all advantages when applied in content-based recommender systems. Nearest Neighbour methods are commonly known as old and reliable go-to methods when nothing else works. It has the worst performance in most cases, but are most likely to succeed[17].

2.4 Demographic Filtering

Recommender systems built around the concept of demographic filtering work very much like their content-based counterpart, although only making use of personal data provided by the users themselves through a registering process, survey response, purchase history etc. This rather than observing and learning user behaviour to classify the users depending their purchase history, ratings etc[20, 42]. A Demographic filtering approach does not, in contrary to content-based filtering, apply a user preference model. It does however apply an item preference model for the items of a system. This basically means that a demographic approach is more privacy-preserving than a content-based filtering one[19].

2.5 Other recommendation techniques

The techniques described under 2.1, 2.3 and 2.4 are the most common when implementing recommender systems [20, 43]. More approaches have been studied and considered, but none that are to be considered more common than the ones described in 2.1, 2.3 and 2.4. One example is using techniques of knowledge-based filtering described in[44].

2.6 Hybrid Systems

A hybrid approach to implementing recommender system means designing the system so as to making use of several recommendation techniques such as for instance collaborative-filtering techniques and content based filtering, making predictions from the combined conclusions of the two. This can be done by unifying the two techniques, by adding features from one into the other or simply by running algorithms for both techniques separately and then combine the results in some way. The most common example of a hybrid based recommender system is the one used by Netflix. While an environment

like Netflix is well suited for a hybrid recommender system, it does not fit everywhere. Why Netflix is considered a hybrid system:

- *Collaborative Filtering:* Observing the watching and browsing habits of similar users.
- *Content-Based Filtering:* Observing users with equal preferences and how they rated certain movies.

2.6.1 Identifying a Hybrid Recommender System

It might not always be a trivial task to identify hybrid recommender systems. Generally hybrid systems do not face the same challenges, described in 2.7, as native implementations do, which is why they are desirable. One should also note that a hybrid recommender system does not necessarily consist only of different base methods such as Collaborative and Content-Based Filtering. A recommender system is of hybrid type if it is based on several different techniques from the same base method as well. That is a system built up on different techniques solely from, for instance, Content-Based filtering is considered a hybrid system as well.

2.7 Challenges of implementing Recommender Systems

There are several challenges a developer of recommender systems might face[37, 38, 45]. Some of them are listed and explained briefly in this section.

2.7.1 "Grey/Black sheep"

Some users might be "grey-" or "black sheeps". Grey sheep refer to users whose characteristics do not overlap with any other group of users, that is not consistently agreeing or disagreeing with any other group of users. Black sheep on the other hand refer to users with extremely varying taste patterns. Something that make predictions nearly impossible.

2.7.2 Sparsity and Subjective Problems

The sparsity problem applies in collaborative filtering based system when it is hard to find items rated by enough people to be feasible to consider (The item-user Matrix is very large and sparse). This is common as the amount of items in most recommender environments exceeds the amount a user is able to explore by far. However sparsity problems in different forms might also be encountered in systems based on content-based filtering. Algorithms for that technique cannot see information as objective or subjective, meaning it cannot distinguish between for instance irony and actual opinions. Which in turn means that a learning algorithm might draw faulty conclusions.

2.7.3 Shilling attacks

In recommender systems where actual ratings constitutes the core of predictions, it is often necessary to introduce precautions to discourage manipulation attempts. These kinds of manipulations might occur in systems where everyone can make ratings, whereas biased users might give lots of positive feedback for products related to themselves and unjustified negative feedback for their competitors.

2.7.4 Cold Start

In a system based on collaborative filtering, new items with no ratings will most likely not be predicted for anyone until it has been rated by several users. The same applies to new users of a specific system. If users do not have any recorded activities on which to base predictions, most recommendations will most likely be inaccurate. Likewise a content-based filtering system will have issues suggesting accurate recommendations of items that the behaviour of a user do not provide evidence for. Additional techniques need to be added to give the recommender system capabilities for solving this.

2.7.5 Lack of computational power

A common challenge when implementing algorithms for collaborative filtering-based recommender systems is the lack of computational power on a single machine whereas most will suffer from scalability problems. Something that is easily understood by imagining that there are systems containing millions of items and even more users, meaning that even algorithms with polynomial or even linear time complexity will be slow. For instance in online environments where recommendations need to be done in real time, without any offline computations, and 2-3 seconds are already too slow.

2.7.6 Equal items with different names

In many systems with millions of items it is pretty common that the same or similar items exist several times in the database, but still have different names or even same names but different ids. Unless thought about, this could be a problem in many recommender systems as they would be treated as different items.

2.7.7 The Exploration VS Exploitation dilemma

A returning issue when working with implementing recommender systems is the exploration versus exploitation trade-off which can be seen mostly in machine-learning environments and is something that every developer needs to consider [46]. Exploration is the task of acquiring new knowledge about an environment or setting while exploitation means using existing knowledge to make decisions or predictions. The dilemma consists in balancing these tasks over time i.e. is it worth risking more exploration for a better reward rather than exploiting already sufficient knowledge about the environment and thereby already getting a good reward with some probability.

And what does this mean when implementing recommender systems for use within e-commerce? An example would be if, let us for simplicity, say that an algorithm used by a recommender system has found one or perhaps a few products that get sold with a high probability every time they are recommended to some type of users. This probability is likely to decrease over time, but how does the algorithm know when to stop recommending the products with known decent profit in favour of exploring new products?

3

Multi-Armed Bandit Algorithms

The multi-armed bandit problem describes how a gambler is standing in front of multiple slot machines (one-armed bandits), needing to make decisions regarding which arm to pull with the intention of maximising the profit. In the sense of recommender systems, this can be modelled as an actor or algorithm having multiple models to choose from, whereas the algorithm needs to decide which model to choose in order to give the best recommendation or make the best prediction.

3.1 Background

The Multi-Armed Bandit problem is a decision-making problem for deciding on a model to use when making predictions about the future[7]. It originates from the Markov decision process and was first formulated in 1952[8], however its concepts have documented discussions from earlier and it at this point in time was not spoken of as Multi-Armed Bandits. Since then there have been numerous different algorithms for how to solve it with good performance.

Some of the bandit models' practical use-areas have, for instance, consisted in providing predictions regarding which projects that are most likely to be successful within different research areas. This helps when determining how investments should be made to maximise the profit. It can also provide useful information regarding failing projects, so as to stop the flow of money in an early stage. Other practical applications for the bandit model have been clinical trials and adaptive routing, where it was used for optimisation.

3.2 The Algorithm

A general idea of The Multi-Armed Bandit problem is as follows:

1. Each arm is to be modelled as a lever, which upon being pulled, provides a reward $r \in \mathbb{R}$ independently, from its own distribution in a setting where all arms have their own distributions.
2. Mean values $\mu_1, \mu_2, \ldots, \mu_n$ can then be computed, where n is the cardinality of the set of distributions for all arms.
3. An agent plays on one lever at a time iteratively and observes the associated rewards for each arm, respectively.
4. The objective of the agent is to maximise her winnings, the sum of the collected rewards.

Within decision theory and probabilistic modelling, regret bounds are often used for measuring negative emotion experience. That is when learning how a different course of action would have resulted in a more favourable outcome. It is thereby desirable to implement algorithms with as low regret bound as possible.

By modelling the Multi-Armed Bandit problem as a one-state Markov decision chain, the total regret ρ after m rounds can be seen as the expected difference between the sum of total rewards from an optimal solution and the sum of the actual collected rewards

$$\rho = m\mu^* - \sum_{i=1}^{m} \widehat{r}_i \tag{3.1}$$

where $\widehat{r}_i$ is the reward of the ith iteration.

Algorithm 1: Multi-armed bandit algorithm

```
   Data: A: the arms, P: the purchases
 1 previousPurchases ← {}
 2 forall the p in P do
 3 |   a ← policyLogic(A, previousPurchases) // Choose an arm depending on
   |   policy
 4 |   ListOfItems ← pullArm(a)
 5 |   if p contains ListOfItems then
 6 |   |   reward ← 1
 7 |   else
 8 |   |   reward ← 0
 9 |   end
10 |   updateArm(a, reward) // Update the arm depending on success or failure
11 |   append(previousPurchases, p) // Append purchase p to the previous
12 end
```

policyLogic returns an arm a based on the policy logic described in section 3.3. *pullArm* returns a list of items that a wants to recommend. *updateArm* is then updating a depending on whether the recommendations were successful or not. A recommendation is

considered successful if a recommended item is purchased by the specific user at a later point in time. The update function is implemented differently for different implementations of the algorithm but the most common way is having α and β variables that get updated with success or failure. These parameters are then taken into consideration when *policyLogic* is executed.

3.3 Policies

The main problem, using Multi-Armed Bandit algorithms, is to maintain a good ratio between exploration versus exploitation, something that has been proved necessary to consider in order to get as high cumulative profit as possible.

3.3.1 ϵ-greedy

The ϵ-greedy[47] policy works by having a fixed value, ϵ, that determines how much the algorithm should explore and/or exploit. The policy starts with generating a random number x. If the randomly generated value x is below ϵ it chooses an arm at random, else it takes the arm with the best performance so far.

Algorithm 2: Multi-armed bandit algorithm with the ϵ-greedy policy

Data: A: the arms, P: the purchases, β_a: tries of arm a, α_a: successful tries of arm a

Initialise all α_a and β_a to 0
forall the p *in* P **do**
  $x \leftarrow random[0,1]$
  if $x < \epsilon$ **then**
    $a \leftarrow random(A)$
  else
    $a \leftarrow argmax(\frac{\alpha_a}{\beta_a})$
  end
  ListOfItems $\leftarrow pullArm(a)$
  if p *contains ListOfItems* **then**
    $inc(\alpha_a)$
  else
    $inc(\beta_a)$
  end
end

3.3.2 Upper Confidence Bound

Upper confidence bound does not, in contrary to the ϵ-greedy policy, continuously explore throughout the whole running time of the algorithm with a fixed probability. Instead it has a discovery phase, where it tries each arm once. After the discovery phase is finished,

it generates the policy for choosing an arm as

$$\mu_a + \lambda\sigma_a \tag{3.2}$$

where μ_a is the estimated reward for arm a, and σ_a a confidence bound measuring the accuracy of our estimate, mean μ_a. The λ is, just as when using the ϵ-greedy policy, a fixed parameter to balance exploration versus exploitation.

A very simple and easy implementation, UCB1[48], can be performed by setting $\mu_a = \frac{\alpha_a}{\beta_a}$ and $\lambda\sigma_a = \sqrt{\frac{2\cdot\ln\sum\beta}{\beta_a}}$ where α_a is the number of times a specific arm has succeeded in predicting a future purchase of an item, and β_a is the number of times the arm has tried to predict items in total.

Algorithm 3: Multi-armed bandit algorithm with the UCB1 policy

Data: A: the arms, P: the purchases, β_a: tries of arm a, α_a: successful tries of arm a

Initialise all α_a and β_a to 0
forall the p *in* P **do**
  $a \leftarrow argmax(\frac{\alpha_a}{\beta_a} + \sqrt{\frac{2\cdot\ln\sum\beta}{\beta_a}})$
  ListOfItems $\leftarrow pullArm(a)$
  if p *contains ListOfItems* **then**
    $inc(\alpha_a)$
  else
    $inc(\beta_a)$
  end
end

3.3.3 Thompson Sampling

A policy on which a lot of recent focus have been put is a policy known as Thompson Sampling [25, 26, 49, 50, 51], which unlike the UCB1 policy goes back to the exploration phase faster when a previously well performing arm starts failing. It is also different in comparison with the so called ϵ-greedy policy which constantly keeps on exploring by some fixed factor. Using Thomson Sampling exploration is instead decaying over time, as the algorithm finds one or more arms that outperforms the others. If, however, there is a change in the market, and a previously good arm suddenly starts to fail more often, the method of Thompson Sampling will once again take a more exploratory approach as

explained above.

Algorithm 4: Multi-armed bandit algorithm with Thompson Sampling without decay

Data: A: the arms, P: the purchases, β_a: tries of arm a, α_a: successful tries of arm a, B: beta-distribution function, λ: decay-factor

```
1  Initialise all α_a and β_a to 1
2  λ = 1.0
3  forall the p in P do
4  |  forall the a in A do
5  |  |  s_a ← B(α_a, β_a)
6  |  end
7  |  a ← arg max (s_a)
8  |  ListOfItems ← pullArm(a)
9  |  β_a ← β_a · λ
10 |  α_a ← α_a · λ
11 |  if p contains ListOfItems then
12 |  |  inc(α_a)
13 |  else
14 |  |  inc(β_a)
15 |  end
16 end
```

The goal of the Multi-Armed Bandit algorithm is to determine on which model to rely, that would work best for the system at a specific point in time. Having the algorithm choose among different statistical models, depending on what seems to be best at this now, takes care of the problem where a specific method or formula which seemed to work great one month might fail horribly the month after. This because of the algorithm always choosing the best out of all available models, depending on actual premises from this specific point in time. To be able to achieve this setting, the system needs to be self-updating, something that Thompson Sampling is by default. The α and β parameters in the algorithm can easily be adjusted to decay over time, simply by setting a decay-factor λ to less than 1.0. If the intention is to trust old results rather than new, λ can also be set to values higher than 1.0.

3.4 The Contextual Multi-Armed Bandit

The Contextual Multi-Armed Bandit Algorithm is a slight modified version of the original problem [11, 15]. Considering the phased solution in 3.2, for the Contextual Multi-Armed Bandit, another phase is added between step two and step three. In addition to the normal setting of a Multi-Armed Bandit there is now additional information that might influence the choice of an agent regarding which arm to pull. Before actually deciding, the agent now sees an n-dimensional feature vector, from now on referred to as context vector, for each arm. The agent then uses these context vectors together

with the rewards of each arm respectively upon deciding on what arm to pull in the current iteration. The agent's goal is to, overtime, gather enough information about the relation between context vectors and rewards, so as to being able to predict which arm is gonna yield the biggest reward only through observing the features of the context vectors.

The users, items and arms all have unique context vectors. When an item is bought by a user, the context vector is updated for both the user and the item. If the arm that was assigned the task of recommending an item made a successful prediction, the system will update its α, as described in 1, to remember that the current arm had a good context vector and should be used more often.

4

Privacy-Preserving Aspects

Personal integrity is a frequently discussed topic in the connected world. Most new products today come with the feature of being able to use the Internet; phones, clocks, cars, TVs, baby monitors and even entire households etc. There are more devices connected to the Internet than there are people living on the planet earth [52]. Being able to stay connected around the clock every day of the month has shed some light on new ethical dilemmas and issues. Internet started out as a place where its users were more or less anonymous. But despite the efforts in present time, creating techniques so as to be more or less anonymous online, a very small portion of the connected world actually make use of these kind of techniques. Applications on our smart devices ask for much more privileges than what their functionality requires, most of the time. This, however, is the case for most online-based systems in general. People are naive most of the time so as to trust systems and applications with whatever personal information they might ask for. This kind of data can then be sold to companies, authorities or in the worst case, hacked or stolen.

4.1 General Aspects of Personal Integrity

Smart phone applications are a big part of many people's lives. They are used for browsing social medias, calendar and scheduling, public transport, work-out, calorie-tracking, storage, e-commerce and just about anything else. Commonly applications do not ask for any privileges when they are new. However, as the amount of users increase one can be certain that sooner or later the application might ask for extended privileges [53] such as access to files, text-messages, contacts and in some cases even camera and GPS. Most people do not see an issue with this, if it is a good application. In many cases this might not even be a problem. Commonly, regular people, would not complain about an applications performing movie, TV-shows or shopping recommendations despite using all possible private data, as long as the recommendations are valid. In many cases, data

changes hands every now and then. Mostly probably because of being sold. What most people do not think of is the fact that once data hits the Internet, it is out of control. Most of the time one can never be sure where its personal data ends up. The more places containing our personal data, the bigger the chance is of it getting leaked, hacked or stolen[54].

These issues are obviously not limited to smart phone applications. It concerns other applications that possess personal information as well. It could be games, e-commerce systems, normal web sites and systems connected to the web in general. It becomes relevant in e-commerce platforms that make use of complex recommender systems and personalised predictions, as those personalised predictions are based on personal information, actions and behaviour in the system. See Chapter 2 for more about recommender systems.
To this comes ethical dilemmas; People actively choose to provide companies with their personal data through consistent use of their applications and products. From the aspect of the groups possessing this kind of data, what is ethically acceptable to do with it? Could it be sold or distributed freely? See [55] and [56] for further research on this topic.

4.2 A Privacy-Preserving User Approach

There are several techniques, of which most are very easy to use, the regular user can apply to achieve good privacy when browsing the web. Other than common sense such as to be careful and critical, normal methods are through use of proxy servers and virtual private networks. One of the more popular techniques as of last years are through use of onion routing protocols [57].

Onion routing means using layers of encryption where each layer consists of decryption at individual nodes. After decryption is done at each individual node the next destination node, in the chain of nodes, is uncovered. As the message reaches the final destination there should be only one layer of encryption left, which when decrypted provides the original message. Because of each individual node only knowing the location of the immediately preceding and following nodes, the original sender remains anonymous. To create and transmit the message initially, an initiator node selects a set of nodes from a list provided by a leader node. The initiator node then obtains a public key from the leader node to send an encrypted message to the first node in the chain, using asymmetric key cryptography to establish a connection and a shared session key[58].

4.3 Privacy-Preserving Systems

Privacy-Preserving systems is an area where much research has been made the last century[5, 19, 59], but despite this very little of the research is being applied in commercial systems. And this for several reasons such as companies wanting to make use of

personal features to improve their systems and the user-experience as a whole, but also for more controversial reasons as those mentioned in 4.1.

4.3.1 Developing a Privacy-Preserving System

Generally you could describe a privacy-preserving system as a system which is developed so as to actively avoid using or even asking for personal data[19, 59]. Although, privacy-preserving systems can contain various levels of privacy where the highest level of privacy would be total anonymity. Obviously some systems are more privacy-preserving than others. Examples of lower levels of privacy are systems that only make use of user-submitted data or data that a user actively choose to provide the service or system with. Among the lowest levels of privacy are systems that record user-activity and -behaviour of actions performed by users. Personal data that can be used, perhaps by the system itself, to classify users in ways that might or might not be more or less ethically acceptable.

5

Implementation

The implementation consists in a full scale implementation of the Contextual Multi-armed bandit algorithm using Thompson Sampling, data extraction, construction of context vectors as well as algorithms for evaluation. Because of each implementation of a Multi-Armed Bandit algorithm being unique depending on the actual use-area and dataset, this chapter will focus on an implementation for e-commerce, and more specifically the Junkyard dataset. The implemented recommender system is built exclusively with Multi-Armed Bandits with its associated techniques having a user experience-based approach rather than a money-making one. The repository for the full code-base can be found at `https://github.com/FredrikEk/Master-Thesis-2015-Contextual-Multi-Armed-Bandits`.

5.1 Approach

As can be seen in chapter 2 the diversity of recommender systems with methods and algorithms for implementing them are huge and rather complex. The implementation performed in this thesis takes a user-centric approach taking most influences from a content-based filtering approach. Due to the algorithmic and machine-learning nature of bandit algorithms, thinking in the ways of content-based filtering when applying methods of Contextual Multi-Armed bandit algorithms comes naturally.

5.2 Thompson Sampling

The implementation of Multi Armed Bandits in this thesis uses Thompson Sampling as the arm learning-policy. It helps the algorithm to update which arm being the best at a specific point in time on-the-fly while being able to predict different recommendations even without any offline calculations. In contrary to the UCB method, Thompson Sampling generate items randomly, which means that it could have an advantage in

e-commerce systems where it would be bad if the same set of items was returned repeatedly until a recalculation is made. It is however an open question, whether this is the case or not, that needs to be evaluated with fixed static data, as is the case for this thesis. Something that could be done using *synthetic data modelling* (7.2.1), but that have not been done in this thesis. The UCB method requires an offline recalculation before new items are recommended. Why this is bad is extra obvious in a setting where a web page might get updated every time a user navigates to a new view, as using Thompson Sampling, would mean generating different items each time a page is updated/reloaded and the algorithm thereby has a higher chance of predicting something that the user finds interesting and perhaps even wants to buy.

5.3 Algorithm

For each purchase (session) the algorithm initially draws a random sample θ from a beta distribution, using each arm's α- and β- variables. The arm arm with highest θ will be the arm used to perform the prediction for this session. In order to find the top ten items, considered the best by arm, the algorithm uses the contextual vector of the current user $user$ together with all the items in the dataset. The contextual vector of $user$ is element-wise multiplied with the contextual vector of an item $item$, which in turn is scalar multiplied with the contextual vector of arm. The resulting value is a measurement of how good $item$ is for $user$ according to arm. The ten items with the highest measurement values then gets recommended. If any of these items are bought by $user$, the prediction of arm is considered correct and the α parameter of arm is increased by one. On the other hand if the recommended items turn out to be faulty

predictions the β is increased by one, whereas the α is left unchanged.

Algorithm 5: Multi-armed bandit algorithm

Data: A: the arms, P: the purchases, I: all the items, U: all the users

forall the p *in* P **do**
  forall the a *in* A **do**
    $\theta_a \leftarrow BetaDistribution(\alpha_a, \beta_a)$
  end
  $arm \leftarrow$ arg max (θ_a)
  $user \leftarrow p_{user}$
  forall the *item in* I **do**
    $item_{reward} = arm_{context} \cdot (user_{context} * item_{context})$
  end
  ListOfItems $\leftarrow$ top10 arg max $(item_{reward})$
  $itemsBought \leftarrow p_{items}$
  if *itemsBought contains ListOfItems* **then**
    $reward \leftarrow 1$
  else
    $reward \leftarrow 0$
  end
  $updateArm(arm, reward)$ // Update the arm depending on success or failure
  $updateItems(itemsBought, user)$ // Update the items which were bought
  $updateUser(user, itemsBought)$ // Update the user who bought items
end

For more information on how the context vectors, $arm_{context}, user_{context}, item_{context})$ work and are built up see section 5.5.

5.3.1 Choosing the number of arms

In the implemented application of this thesis, the algorithm use 50 arms. A good unofficial measurement that has been found experimentally in this project in combination with studies of current literature is that the number of arms should exceed the total number of feature-elements in the arm context vector. It was noticed that using too few arms could mean not generating good enough variance of the elements, while using too many arms would hurt the performance, in terms of successful recommendations, when using small datasets.

5.4 Data extraction

The data used in the implementation is extracted from an MSSQL database containing the entire Junkyard dataset. The extracted data that are used for training, recommendations and validation consists of three SQL- views. Each containing context of users, items and orders respectively.

5.4.1 The Userdata view

The query that builds the userdata view can be viewed in the appendix, listing A.1. The columns UserId, ZipCode, DateOfBirth and Gender in the resulting table are used when constructing the context vector of each user (see more under 5.5.1) whereas the column Updated is used to check when the specific user last updated his profile. Most of the time it is used for notifying when a user changed it's domicile i.e the ZipCode changed. Something that is important because one of the features of a user's context vector is domicile.

5.4.2 The Itemdata view

The query that builds the itemdata view can be viewed in the appendix, listing A.2. The columns ItemId, Gender and category in the resulting table are used when constructing the context vector of each item (see more under 5.5.2) where as the columns ItemCreated, ItemModified and CurrentStock are used to check the saldo of an item i.e if the saldo is at zero at a specific point in time, when a recommendation is about to be made and we know that the saldo wont increase in the near future, then obviously we do not want to recommend that item as it would contribute to a bad user experience.

5.4.3 The Orderdata view

The query that builds the orderdata view can be viewed in the appendix, listing A.3. All of the columns are used to train the arms of the bandit algorithm, from one point in time to another. This view is also, however, used for validation. By chronologically observing the data, "future" data can be used as validation. Simplified, the algorithm looks at which user bought what item and at what point in time. More about the training part of the algorithm can be found under 5.6.

5.5 Context vectors

Each user, item and arm has it's own context vector. The vectors of each arm are generated randomly, whereas the Thompson Sampling algorithm determines which arm is the best, currently. The user and item vectors are constructed using information provided by users upon registering, by their history of purchases and from their behaviour in the system, whereas itemdata gets added when administrators of the system add new items to the database.

In the following sections the context vectors are explained in more detail. The number of elements each feature is represented by are shown in Table 5.1.

Feature	Number of Elements
Gender	2
Category	33
Age	3
ZipCode	1
Popularity	2
Total	41

Table 5.1: Feature Data

The representation of *Gender* is done using one element each for male and female respectively. The *Category* feature has one element for each category. *Age* is represented by three age spans; young − the buyer is younger than 18 years old, middle aged − the buyer is between 18 and 33, old − where the buyer is older than 33. *ZipCode* is represented by only one element consisting of the three first digits in the area-number. The *Popularity* feature contains two elements; most bought last week and most bought last month.

5.5.1 The User Context Vector

The implementation of the user context vectors follow this simple model:

$$\begin{cases} \text{Gender} = \text{Given by the user/database, can be male or female} \\ \text{Category} = \text{By observing the category of purchases from a specific user} \\ \text{Age} = \text{Given by the user/database} \\ \text{ZipCode} = \text{Given by the user/database} \\ \text{Popularity} = \text{By observing if the user buys popular items or not} \end{cases}$$

5.5.2 The Item Context Vector

The implementation of the item context vectors follow this simple model:

$$\begin{cases} \text{Gender} = \text{Given by the admin/database, can be male, female or both} \\ \text{Category} = \text{Given by the admin/database} \\ \text{Age} = \text{By observing the age span of users who bought this item} \\ \text{ZipCode} = \text{By observing the domicile of users who bought this item} \\ \text{Popularity} = \text{By looking at the most bought items from the last week and the last month} \end{cases}$$

5.5.3 The Arm Context Vector

The purpose of the context vectors of the arms is to control how much weight to put on the different features in the context vectors of items and users when computing their scalar product. In the initialisation phase of the algorithm, the feature-values of the arm's context vectors are generated randomly from a continuously uniform distribution, to values between 0 and 1. After which they never change.

$$\begin{cases} \text{Gender} = \text{Randomly generated} \\ \text{Category} = \text{Randomly generated} \\ \text{Age} = \text{Randomly generated} \\ \text{ZipCode} = \text{Randomly generated} \\ \text{Popularity} = \text{Randomly generated} \end{cases}$$

These vectors are needed in the Contextual Multi-Armed Bandit setting as described in 3.4.

5.6 Training elements

As mentioned in 5.1 the implemented set of algorithms are based on a content-based filtering approach, using learning techniques for "arms", items and users in the setting of Multi-Armed Bandits. The method used for the training of arms are called Thompson Sampling and can be viewed under 5.2. The training of user profiles and items follow a user preference model [60]. The training consists in updating the features of the context vectors.

5.6.1 Training of Arms

The context vectors of the arms are initialised with random values as described under 5.5.3 and training the arms only consists of making sure we use the better arms more often.

5.6.2 Training of User profiles

Assuming that a specific user is likely to buy products with equal features in the future, user profiles progress over time and learn the features of items that are bought by the specific user. Item features contributing to this learning process are category, gender and popularity. The structure of the user context vector are shown in 5.5.1. User profiles get initial training prior to running the algorithm.

5.6.3 Training of Items

Items learn the preferences of the users who purchase them over time, as shown in 5.5.2. As several of the item features rely on classifications made through learning user-preferences, items get initial training prior to running the main algorithm.

5.7 Frameworks for implementing recommender systems

As initially described in Chapter 1 this project uses frameworks for implementing recommender systems. Both of the used frameworks come with a well-documented java API, which make them really intuitive and user-friendly. The plan, initially, was to use frameworks for redundant efforts such as implementing data structures, basic algorithms and visual tools etc. This have been true to some extent, but in most cases implementations have been made from scratch.

5.7.1 Apache Mahout

Apache Mahout is a scalable machine learning library and framework for implementing recommender systems [2]. The core of the implemented application in this thesis is written from scratch using data structures and some algorithms from Mahout; Through use of Mahout, vectors and matrices with their algebraic operations could be used of the shelf. Mahout also contained some built-in basic machine-learning libraries such as one for beta distributions, used in this project.

5.7.2 LensKit

LensKit is, like Mahout, a framework for implementing recommender systems with support for some of the more common algorithms[1]. Similarly with Mahout, the implementation of this thesis only make use of some data structures from LensKit.

5.8 Implementation of Collaborative Filtering for Evaluation

As can be read in Chapter 2, the mainstream approach of implementing recommender systems [61] to date is through use of collaborative filtering- based techniques, which is why it makes sense to use one for comparison. The implementation of collaborative filtering in this project is performed taking a user-based approach, meaning that similarity is computed between users rather than items. More about user-based collaborative filtering can be found under 2.1.1. The implementation structure look like any other Collaborative Filtering-based approach, using a similarity computation followed by a prediction generation.

5.8.1 Similarity Computation

The standard implementations of collaborative filtering based techniques use ratings as described in 2.1. All systems do not however make use of ratings. For instance, it is rather trivial to realise why using ratings would not be feasible for garment-based e-commerce, as is the case for the dataset used in this thesis; the Junkyard e-commerce platform does not make use of ratings.

Our implementation follows a model described in [62], where similarity of users are computed using the Jaccard Distance of purchases. The Jaccard Similarity is computed as:

$$J(u_1,u_2) = \frac{|u_1 \cap u_2|}{|u_1 \cup u_2|} \tag{5.1}$$

Where, in our implementation, u_1 and u_2 are vectors each belonging to different users, whose similarity is to be computed. These vectors contain Boolean values for all items in the database, determining whether a specific item has been bought by this user or not. Computing the Jaccard Distance as explained above will return a value $0 < J(u_1,u_2) < 1$ and can thus be seen as "How similar is user u_1 to user u_2?".

5.8.2 Prediction Generation

After the similarity is computed between a user u and all other users $u \in \{u_1, u_2, \ldots, u_n\}$ who bought items, the algorithm chooses which items to recommend using the following formulae:

$$S_i^k = \sum_{i \in u_j, j \neq k} J(u_j, u_k) \tag{5.2}$$

$$P(u_k) = top\ 10\ S_i^k \tag{5.3}$$

Where the resulting S_i^k is the value of item i, where i is an item that user k has not bought. J is the similarity function for a user u_j and a user u_k described in equation 5.1. $P(u_k)$ is a vector containing the top ten items for user u_k.

5.9 Recommender Baselines for Evaluation

Aside from the implemented, significantly different algorithms used for evaluation described in 5.8, some basic statistical models have been implemented to be used as a kind of baselines throughout the the project. A baseline is a static algorithm for making recommendations, following some statistical model. Some examples of implemented baselines are (strategy):

- Always recommend the most sold item, as of the last hours, days, weeks, months or season. (People always buy the most popular items)

- Always recommend items based on gender. (Men and women always buy the same items as other men and women, respectively)
- Always recommend items based on age. (People at a certain age always buy same items as other people in the same age)
- Always recommend items based on domicile. (People from city X always buy the same items as other people from city X)

Baselines have been used before any other extensive evaluation method, something that has been of great help when implementing the application, as these baselines can be seen as evaluation in its most basic form.

6

Result

Multi-Armed Bandit Algorithms are rarely used in recommender systems for e-commerce. In this chapter follows a presentation of the results achieved in this project using our own implementation of The Contextual Multi-Armed Bandit Algorithm applied to the a dataset from a live e-commerce system. This chapter will focus on the actual results. For evaluation and discussion regarding how good the results were, see Chapter 7.

6.1 Recommender System

Taking a privacy-preserving approach, recommendations performed by the implemented application can be split into three categories or levels of context which can be seen under 6.1.1, 6.1.2 and 6.1.3. In the following series of plots, the results of the implemented application using Contextual Multi-Armed Bandits are illustrated. Each subsection contains four graphs illustrating different time periods and for each subsection the data is limited taking a more privacy-preserving approach. The graphs show a percentage of correct predictions on the y-axis and the total number of predictions made on the x-axis.

6.1.1 Using all available context

The lowest level of privacy consist in using all available context, meaning that the algorithm takes everything from user-submitted data, to observed and learned user-behaviour and -history, into account when predicting purchases. Here follow some graphs with different time aspects, visualising the performance with number of predictions on the x-axis and correctness in percent on the y-axis using this policy.

Graphs in figure 6.1 show all arms relative to the best arm. The best arm is chosen as the one with best performance posterior of the simulation. Notice that ”All arms” follow the same pattern as the ”Best arm” towards the end. This is due to the fact

that the algorithm is more or less done with the exploration phase and has started the exploiting phase, choosing the best arm over and over.

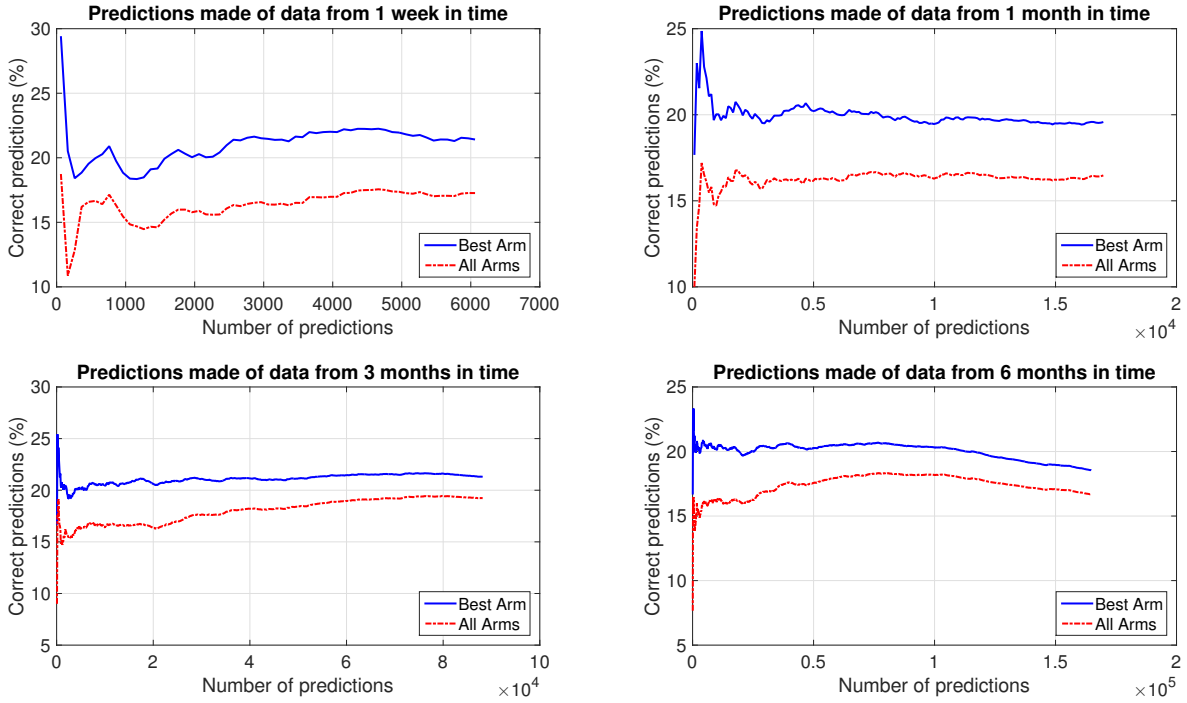


Figure 6.1: Results using all available context

6.1.2 Using only user-submitted data

The middle level of privacy consist in only using data submitted by the users of a system. Data that they agreed to submit upon registering an account or purchase in the system. Recommendations are done using only this data, without having the algorithm train on observed user behaviour in the system. Here follow some graphs with different time aspects, visualising the performance with number of predictions on the x-axis and correctness in percent on the y-axis using this policy.

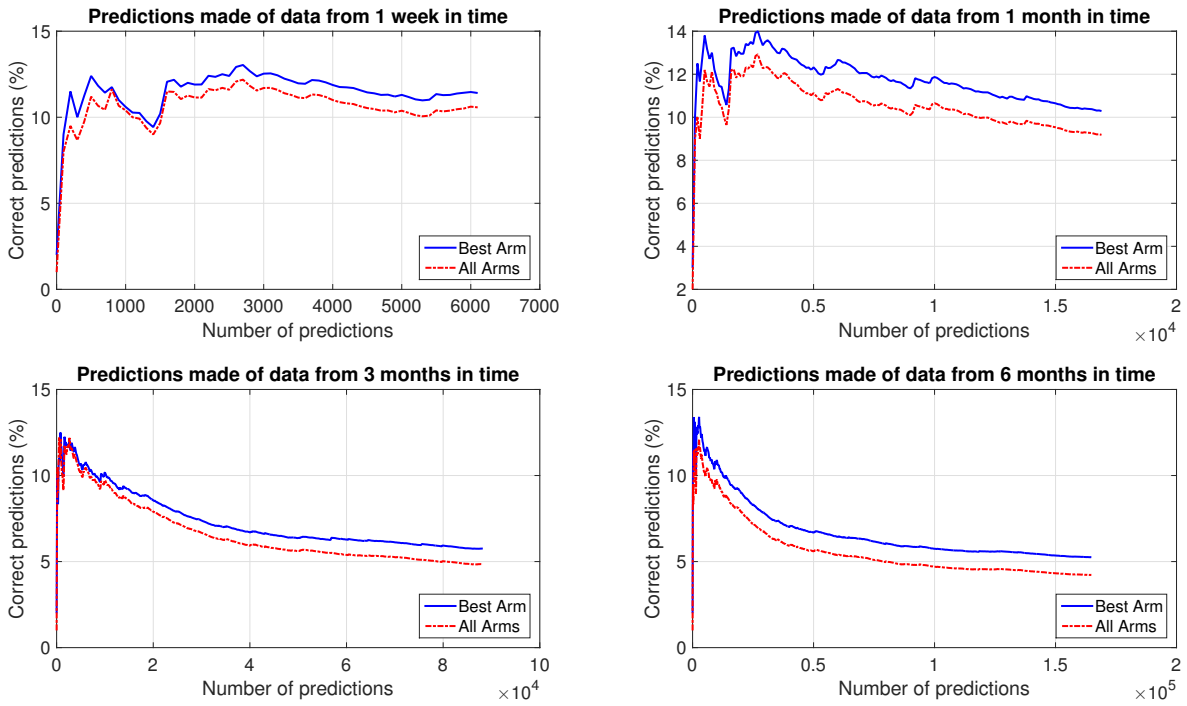


Figure 6.2: Results using only user-submitted context.

6.1.3 Using anonymous data

The highest level of privacy consist in total anonymous data. Recommendations are generated only through observing the flow of products, what are being bought and when.Due to the nature of Contextual Multi-Armed Bandit Algorithms, the scenario of anonymity can only be applied to a certain extent with the dataset used in this thesis. One way of making anonymous recommendations can be seen as recommending the most popular items from a certain time period. Results of this can be seen under 6.3.1.

6.2 The Application

The implemented application is implemented in a scalable manor but without any graphical interfaces, meaning that the core of the application is easily modified to use more or less features, arms, training sets, predictions etc. Inputs are accepted following a simple model, as can be seen in 5.4, meaning input could be taken from other systems with some slight modifications. The application also includes support for automatic logging and plotting of the results. Everything to make the application as easily understandable and usable as possible.

6.3 Algorithms for Evaluation

The implemented algorithms for evaluation and comparison consist in two different approaches. The first one is through use of simple statistical models, and the second one is a full-scale implementation of a User-Based Collaborative Filtering Algorithm.

6.3.1 Simple Statistical Models

Simple statistical models have been implemented, as described under 5.9, and used as baselines while implementing the main application, something that have been of great help when trying new methods early on in the implementation steps. It is however pointless to use baselines with poor performance in the final result, which is why the following graphs only show the baseline with the best performance. This baseline is a statistical model based on the purchases from the last week of any point in time. It will always recommend the ten most bought items as of the last week, to all users. The following graphs got the same layout as the ones in 6.1, namely with a percentage of correct recommendations on the y-axis and the total number of recommendations made on the x-axis.

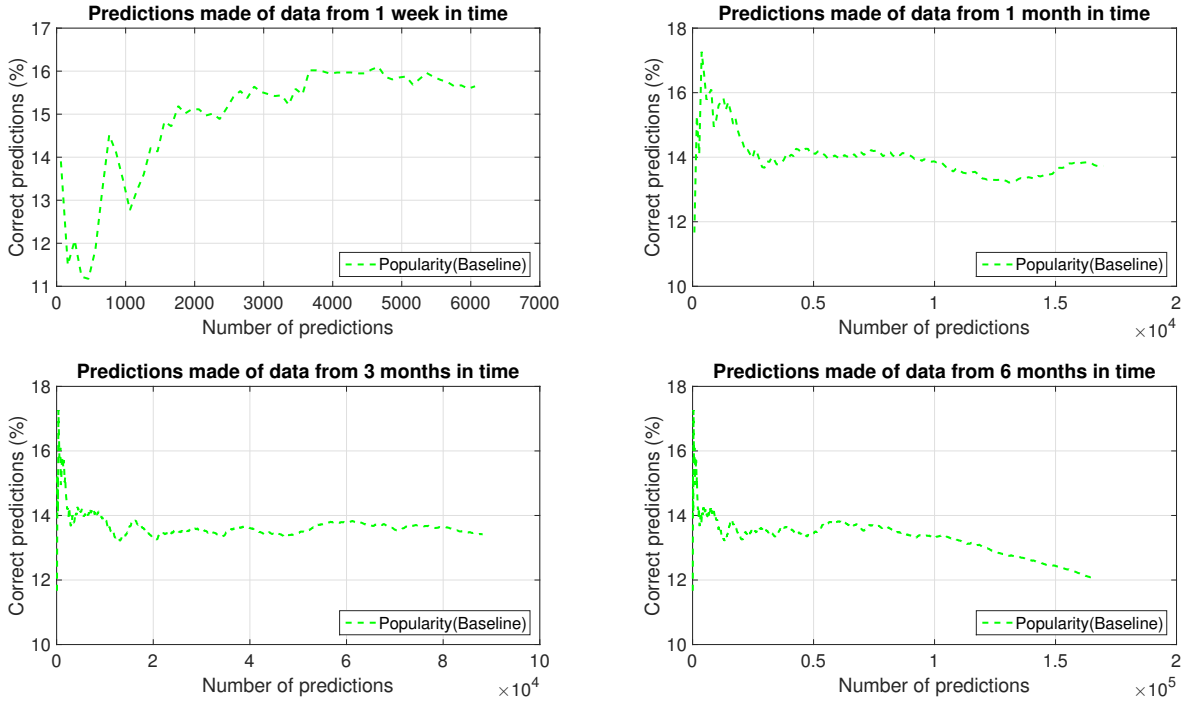


Figure 6.3: Graphs showing recommendations based solely on popularity following the baseline that recommends the most bought items as of the last week

6.3.2 User-Based Collaborative Filtering

A full-scale implementation of an application using collaborative filtering-based algorithms have been implemented as described under 5.8. Here follow some graphs of its performance where the graphs got the same layout as the ones in 6.1 and 6.3.1, namely with a percentage of correct recommendations on the y-axis and the total number of recommendations made on the x-axis.

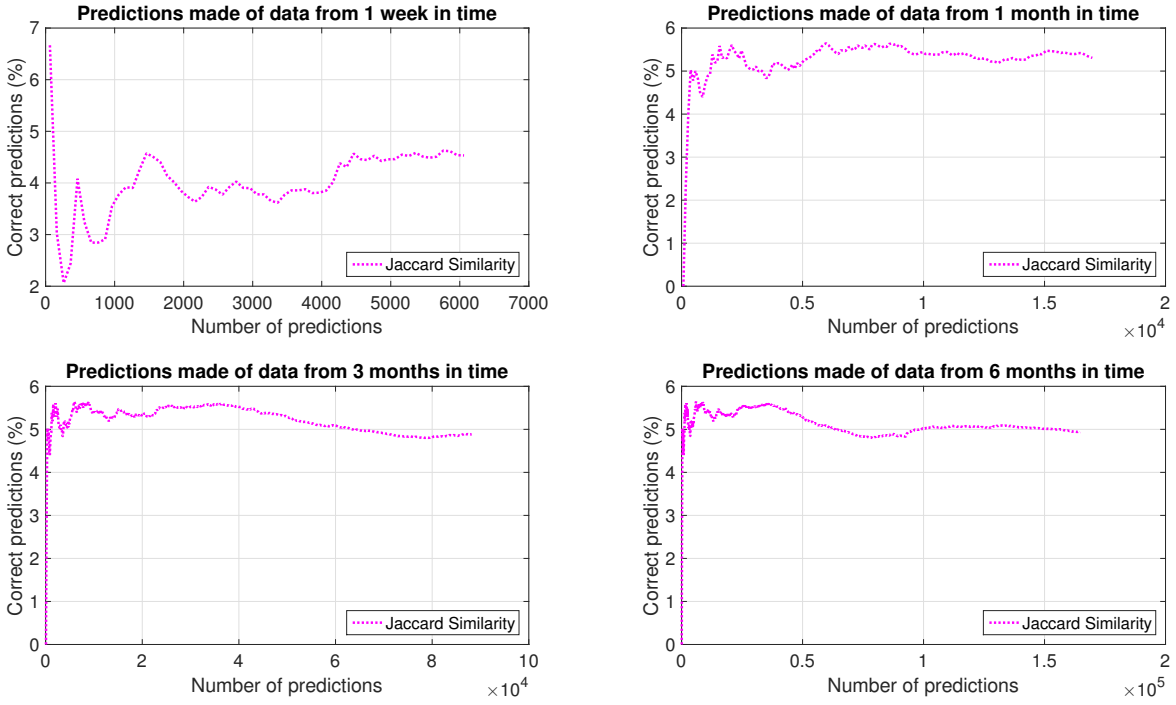


Figure 6.4: Graphs showing result when recommending items based on the jaccard similarity between users.

7

Discussion

As stated in the introduction, this thesis aims to evaluate the performance of Multi-Armed Bandit Algorithms in an e-commerce environment. In this chapter evaluation of user-experience and comparisons with other algorithms will be presented together with thoughts regarding the result and future work.

7.1 Evaluating The Result

The result can be evaluated depending on several different aspects. In this section focus will be put on evaluation of performance and user-experience as well as some thoughts regarding why the result turned out the way it did.

7.1.1 Performance

The percentage of correct predictions can be seen on the y-axis of the graphs in section 6.1. By comparing the graphs in figure 6.1 one can draw the conclusion that the best recommendations are made in the time interval between one week to one month posterior to starting the application. Predictions in this period got an accuracy of 19 to 21 percent. Seeing how this is in a garment-based e-commerce environment it makes sense that new collections of products are added every other month, whereas the products from last month's collection do not sell as good. The simulation made in these graphs starts in late spring, 2012-05-01, meaning that recommendations of cloths for the summer are more likely to be of satisfactory than a few months later when summer-based cloths are recommended for the fall or winter. The conclusion here is that if this application was to run in a live system, it would make sense to reset the trained data prior to releasing a new collection and/or before each new season.

As mentioned earlier, the dataset used in this thesis contains only static data. Meaning that for a prediction to be seen as correct, a recommended item has to be purchased at

some point in time after the recommendation is performed. The problem using static data rather than testing an application live in this setting is that possibly good recommendations are not measured, because an item was not purchased. One can never know if the user would have bought more products if products that she could possibly like would be presented to her. Hence the only thing that can be measured is actual purchases, while in theory a user is likely to buy more products based on the recommended items. Something that could be said to be certain about the application here is that it predicts correct items with at least 19-21 percent accuracy and probably higher in most cases.

All the graphs in Chapter 6 expressing anonymous data to some extent, proves that taking a more privacy-preserving approach, actually gives worse performance in terms of measuring successful recommendations. By comparing the graphs in figure 6.1 where the algorithm is using all available context, with the other graphs in figure 6.2 where the algorithm is using less context, it is possible to see a difference of about 6-10 percent depending on which graphs are observed. It is hard to tell if it is worth it or not, but seeing how things work in reality where most people accept any license agreement of applications or web-pages, one can draw the conclusion that as long as the observed data is used in ways that benefit the actual user and not a third party, it is at least socially acceptable. Facebook is a good example of this, which in their licence agreement [63] state that they have all the rights to any data you submit or take part of online, when using their services. They also state that your data will be accessible by companies and third parties of their choosing.
There are also other aspects to a good result than through only measuring the number of successful recommendations, such as user experience and satisfaction level of customers. Unfortunately these are hard to measure without deploying the system live. See more about user-experience under 7.1.2.

So far we have seen that the implemented application can predict shopping behaviour of most users with a probability of around 20 percent. So the question is how well does other similar applications perform on similar datasets. One similar application in a similar setting can be seen in [61] where they use significantly different algorithms and achieve correct recommendations with a probability of about 17 percent. The authors of [61] make use of a collaborative filtering-based techniques, but with significantly better performance than the collaborative filtering-based algorithms used in this project. Nevertheless, the content-based filtering implementation using Contextual Multi-Armed Bandits in this thesis outperforms the algorithms used in [61], something that ought to be considered as successful. In figure 7.1 below, the different implemented algorithms of this thesis is presented with different colours, using the time interval of one month because of its relevance as described above.

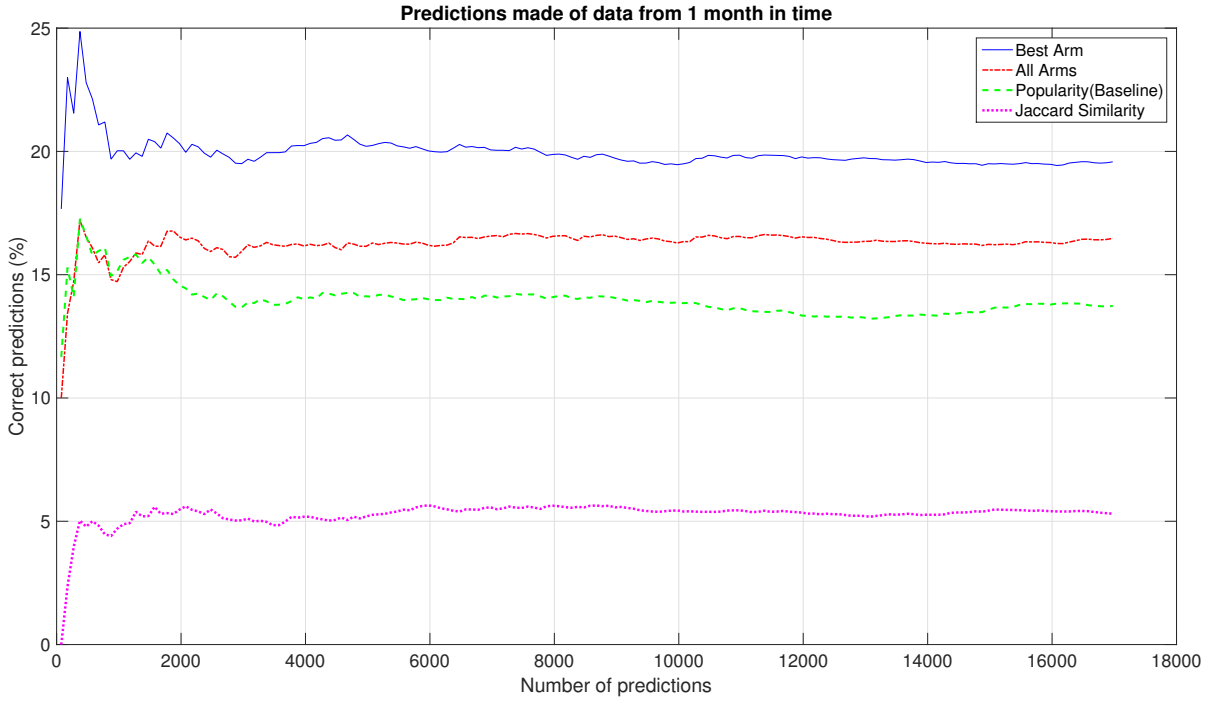


Figure 7.1: Graph showing the results of all implemented algorithms from one month in time

And as can be seen in this graph, the Contextual Multi-Armed Bandit Algorithm outperforms all of the others. Notable is also that the popularity baseline algorithm appears to have pretty high performance. Something that is easily realised why it is the case. This strategy is very bad in other aspects as if taking User-Experience into account. See section 7.1.2 for more details regarding this.
The next graph, in figure 7.2, shows the performance of all algorithms in different colours, measuring performance of purchases over one year.

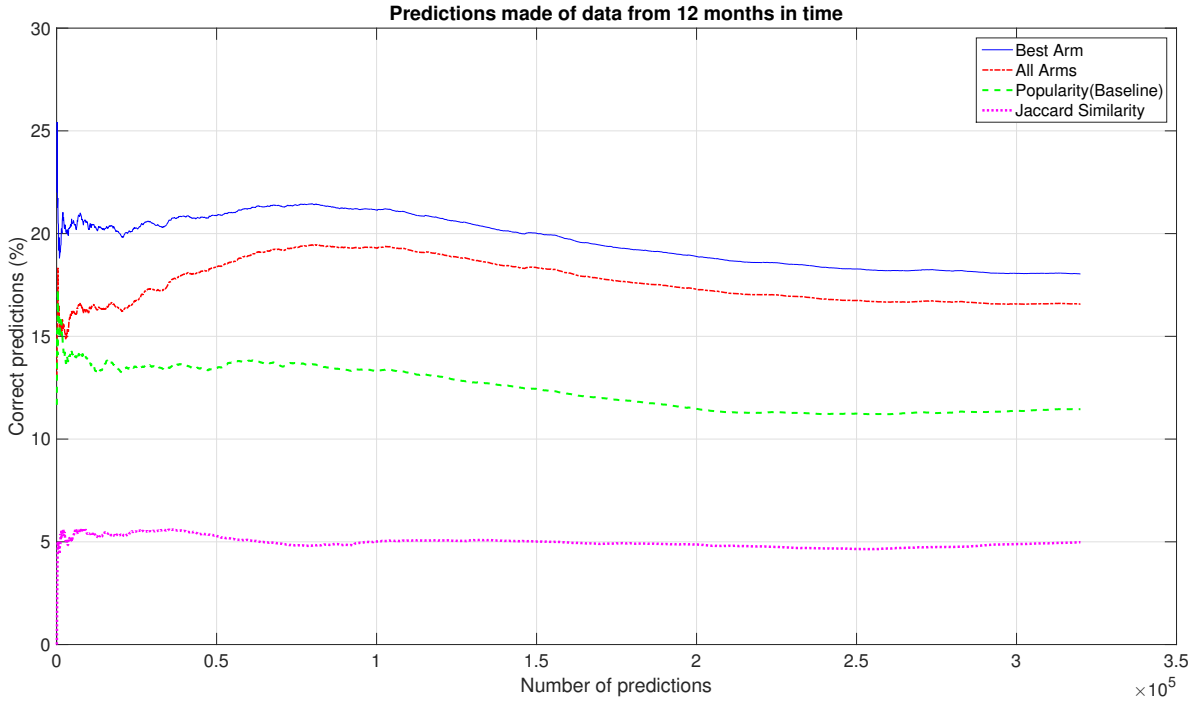


Figure 7.2: Graph showing the results of all implemented algorithms from one year in time

Here one can see that the result does not change even in the long run. All algorithms perform best during the first months due to reasons expressed in the beginning of this section, but even after a long time the order of good versus bad algorithms do not change. Something notable is what happens in the time span between 1 and 3 months, where we can see a dip in performance in terms of successful recommendations. The graph stretches from 2012-05-01 to 2012-07-31. In this time period there is a shift in products wanted by the users. A user that has bought for instance some pairs of jeans during May has a high probability of being recommended jeans during the later months too. However, during the summer it is more likely that the user wants to buy shorts and other products more suitable for use during the summer. As can be seen in graph 7.2, the popularity curve also become worse during these months. This means that user-behaviour is harder to predict during these months. This might be the case either because there is a larger set of items that can be recommended, or because user behaviour is not as predictable during the summer as during the spring. As mentioned initially in this chapter, it is also likely that new collections of garment are released before each season, which might be another contributing factor to the dip in performance during the specific time spans.

One could wonder: why did the collaborative filtering-based algorithm using Jaccard

Similarity perform so poorly with the Junkyard dataset, in comparison with the other algorithms in this thesis and the ones described in [61]. This, of course, might depend on a variety of different reasons such as algorithms or datasets being significantly different. Seeing how the focus of this evaluation is on the performance of Multi-Armed Bandits, further investigation of the poor result of the above mentioned will be left for future work.

As explained in 3.2, regret bounds are often used for measuring negative emotion experience in decision theory and probabilistic modelling. In figure 7.3 below, the total regret is illustrated for 'All Arms', 'Most Buys' and 'Jaccard Similarity' relative to the 'Best Arm'.

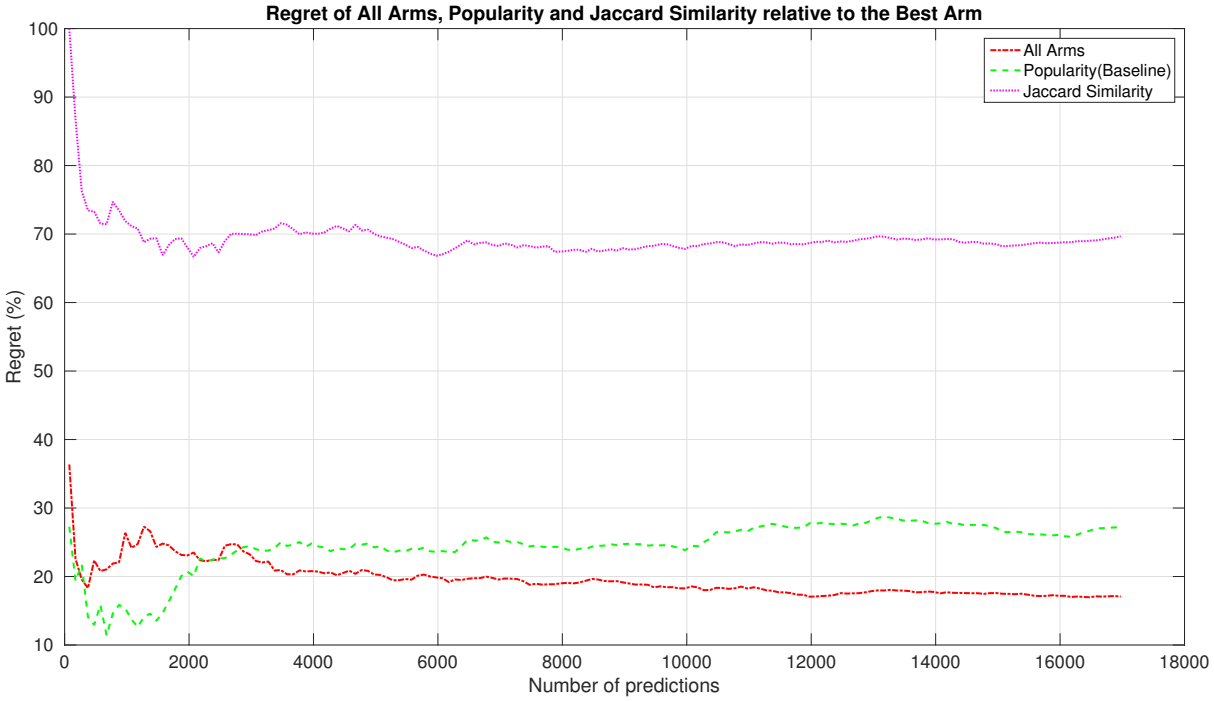


Figure 7.3: Regret of 'All Arms', 'Most Buys' and 'Jaccard Similarity' relative to the Best Arm.

An interesting scenario that was stumbled upon is if the following happens (7.4):

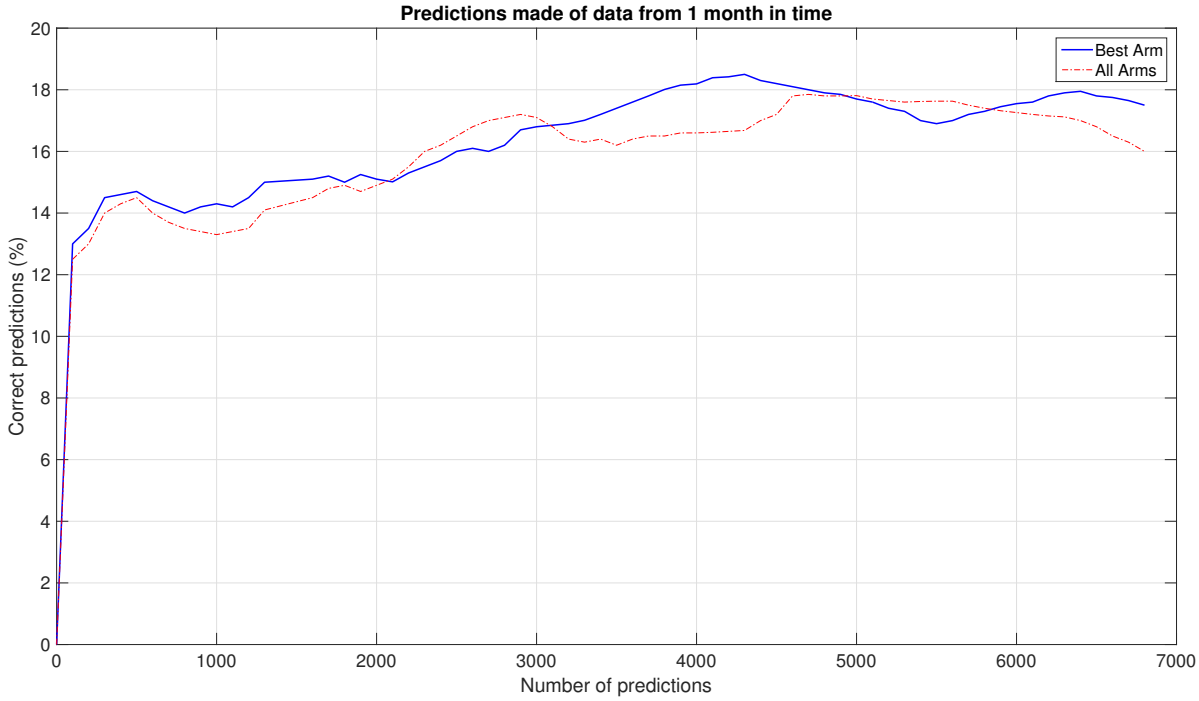


Figure 7.4: All Arms outperforming Best Arm for some time

What can be seen here is running the algorithm for six months, and where All Arms seemingly perform better than the Best Arm at some point in time. There is a simple explanation to this phenomenon. Posterior to running the algorithm, the algorithm looks at all arms to see which one that is having the best performance. Thus the arm with the best performance posterior to running the algorithm is not necessarily the best arm throughout the whole simulation.

7.1.2 User-Experience

User-Experience is more tricky to measure than performance. Despite suggesting items that are likely to be bought statistically, it does not mean that it will make a specific customer happy. Recommending items that might be of no value for a specific customer, but are best-sellers, is unarguably not the best way to present recommendations even if it will show good performance. This can be seen in the graphs in figure 6.3 where recommendations are made entirely based on whats the currently most bought items as of the last week.

Like a physical store

When visiting a physical store, you would not want the shopping assistant to only propose the same items as she is proposing to everyone else. For instance by proposing only the best-sellers as of the last week, not taking anything else into consideration. Instead you would want the shopping assistant to listen to what kind of items you like and have enjoyed earlier, and what you are currently looking for. The same thing applies for most people when browsing E-Commerce websites or platforms. If you browse for or buy certain items, the system should adapt to this and actually try to help you find what you might be looking for in the future. That is if you are looking to buy a pair of shoes and you are recommended ten different shirts, just because the shirts happened to be on sale last week meaning that a lot of people have purchased them, you will not be satisfied or any closer to finding the pair of shoes you were after.

If however, let us say, the shopping assistant was to follow you around noting what you were looking at or even what you were buying in other stores, you would probably find her creepy and probably stay away from her stores. This despite the fact that it might mean ending up with an item you enjoy, faster. Once again, the same idea can be applied in E-Commerce systems; using all data you can from a specific user is not ethically acceptable. It is however more socially acceptable as described in 7.1.1. This could be data such as cookie-based history of other browsed web sites. It could also be data retrieved from buying and selling information gathered in other systems.

Our implementation

When implementing the algorithm in this project, the physical store was kept in mind. Users should not relate to the system as the creepy shopping assistant. The implemented application thus only focus on using data gathered from the platform which it was built for and even there discussing different privacy-preserving aspects. Neither should users relate to the system as the lazy shopping assistant who only gives the same recommendations to everyone. Instead focus was put on making the customers feel that the system or shopping assistant actually proposed valid items that the customer could be interested in. This by giving ten recommendations that match the specific customer's profile.

Taking it live

Observing the shopping history of a user in the store is an intuitively good technique to figure out what the user is interested in. However, if taking the system live,predictions would also based on the current live session. The taste of a user would probably most often be similar from the different times she visits the website, but she might want to shop from different categories during different sessions. Looking for shoes at one time should not mean that the user only gets shoe recommendations the second time she browses the site. If the user however only looks for gender-specific items, most of the time there would be no point in suggesting the opposite.

Best-seller recommendations are not useless in any way. Using this application in a live setting could mean using best-seller recommendations on the front page so that people occasionally browsing the page in hopes of finding something they like might get lucky, while the personalised recommendations could be used on the specific user-pages.

7.2 Future Work

There are still issues related with the result of this project that could be desirable to address, but that goes out of the scope of this thesis. In this section we will present some concrete ideas of possible future work related to this thesis.

7.2.1 Synthetic Data

Synthetic data is, as opposed to authentic data, generated within some behavioural model. It is explained in [64] and described as: ”Synthetic data can be defined as data that are generated by simulated users in a simulated system, performing simulated actions.

One obvious use-area for synthetic data is the possibility of doing realistic testing of a system or parts of a system before deploying them live. There are however many difficulties that need to be considered before implementation can start. Difficulties that are not entirely addressed in current research as it varies in different settings and systems.

7.2.2 Extensive Collaborative Filtering-Based Algorithm

The implemented collaborative filtering-based algorithm in this thesis perform poorly in comparison to what current research expresses when describing collaborative filtering-based algorithms. It would be interesting to make use of different similarity computing techniques to see if it could, first of all, provide a better result than the implemented collaborative filtering-based algorithm in this thesis, but also if it could beat the performance of the Contextual Multi-Armed Bandit Algorithm implemented in this thesis.

8

Conclusion

In this thesis the performance of Contextual Multi-Armed Bandit Algorithms have been measured and evaluated in terms of successful recommendations, user-experience and possibility of working using privacy-preserving methods, in a garment-based e-commerce environment. The result can be seen in chapter 6. In the aspect of making successful recommendations and giving a good user-experience the results look good. But also in the cases where privacy-preserving approaches have been taken, the results are fairly good. Depending on the setting and system, if required, it would thereby be feasible to consider even the privacy-preserving modes of the algorithm as performance in terms of successful recommendations are fairly good despite those restrictions.

8.1 The Research Question

In the introduction we defined the research question as:
Would the algorithmic approach of Contextual Multi-Armed Bandit Algorithms perform well enough to belong among the different algorithmic approaches to consider when implementing recommender systems?
A justified answer to this question, with respect to the results presented in Chapter 6 and the reasoning of the results presented in Chapter 7, is ”yes”. A reasonable conclusion that can be made from observing the result where purchasing behaviour is predicted with a probability of over 20 % is that considering Contextual Multi-Armed Bandit Algorithms when implementing recommender systems is highly appropriate.

The more complex but also most suitable answer to the research question is ”probably” or ”it depends on the setting and environment”. As the name Contextual Multi-Armed Bandit Algorithms indicates, the algorithm makes use of contexts such as a user's context, personal information, and might thus not be eligible in systems where all data is to be totally anonymous or where only non-personal data exists. However, as can be seen

in Chapter 6 results are fairly good despite using less user context, but there might be other algorithms more suitable for this purpose.

Another conclusion that can be made using Contextual Multi-Armed Bandit Algorithms in garment-based e-commerce systems, is when the e-commerce system does not make use of ratings, where otherwise reliable collaborative filtering-based algorithms might not perform so well (see 6.3.2), Contextual Multi-Armed Bandit Algorithms could work better because it observes user-context and -behaviour rather than ratings. Although this thesis is only testing Contextual Multi-Armed Bandit Algorithms using static data from one garment-based e-commerce system, which does not make use of ratings. To be certain that equally good results can be achieved frequently, it would be necessary to test the algorithm on data from more and different systems. Especially significantly different systems, such as ones that does not involve clothing.

Contextual Multi-Armed Bandit Algorithms work in systems with or without ratings. As long as there exist recorded data of users and their behaviour in the system, Contextual Multi-Armed Bandit Algorithms should have good chances of providing good and reliable recommendations in the aspect of user-experience as well as of performing successful recommendations.

Bibliography

[1] J. A. K. Michael D. Ekstrand, Michael Ludwig, J. T. Riedl, Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit., in: Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11). ACM, New York, NY, USA, 2011, pp. 133–140.
URL `http://lenskit.org/`

[2] The apache mahout^{TM} machine learning library. (2015).
URL `http://mahout.apache.org/`

[3] J. B. Schafer, J. Konstan, J. Riedl, Recommender systems in e-commerce, in: Proceedings of the 1st ACM conference on Electronic commerce, ACM, 1999, pp. 158–166.

[4] D. Bergemann, J. Valimaki, Bandit problems.

[5] C. Dwork, Differential privacy, in: Encyclopedia of Cryptography and Security, Springer, 2011, pp. 338–340.

[6] Tyler Lu, Dávid Pál, Martin Pál, Contextual Multi-Armed Bandits (2010).

[7] K. P. Murphy, Machine Learning: A Probalistic Perspective, The MIT Press, 2012.

[8] H. Robbins, Some aspects of the sequential design of experiments [1952], in: Herbert Robbins Selected Papers, Springer, 1985, pp. 169–177.

[9] J. Langford, T. Zhang, The epoch-greedy algorithm for multi-armed bandits with side information, in: Advances in neural information processing systems, 2008, pp. 817–824.

[10] R. Kleinberg, Anytime algorithms for multi-armed bandit problems, in: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, Society for Industrial and Applied Mathematics, 2006, pp. 928–936.

[11] T. Lu, D. Pál, M. Pál, Contextual multi-armed bandits, in: International Conference on Artificial Intelligence and Statistics, 2010, pp. 485–492.

[12] A. Mahajan, D. Teneketzis, Multi-armed bandit problems, in: Foundations and Applications of Sensor Management, Springer, 2008, pp. 121–151.

[13] V. Kuleshov, D. Precup, Algorithms for multi-armed bandit problems, CoRR abs/1402.6028.
URL `http://arxiv.org/abs/1402.6028`

[14] R. Howard, Dynamic Programming and Markov Processes, Technology Press-Wiley, 1960.
URL `http://books.google.ie/books?id=fXJEAAAAIAAJ`

[15] L. Li, W. Chu, J. Langford, R. E. Schapire, A contextual-bandit approach to personalized news article recommendation, in: Proceedings of the 19th international conference on World wide web, ACM, 2010, pp. 661–670.

[16] R. Van Meteren, M. Van Someren, Using content-based filtering for recommendation, in: Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop, 2000, pp. 47–56.

[17] M. J. Pazzani, D. Billsus, Content-based recommendation systems, in: The adaptive web, Springer, 2007, pp. 325–341.

[18] J. L. Herlocker, J. A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 1999, pp. 230–237.

[19] E. Aimeur, G. Brassard, J. M. Fernandez, F. Onana, Privacy-preserving demographic filtering, in: Proceedings of the 2006 ACM symposium on Applied computing, ACM, 2006, pp. 872–878.

[20] M. J. Pazzani, A framework for collaborative, content-based and demographic filtering, Artificial Intelligence Review 13 (5-6) (1999) 393–408.

[21] G. Linden, B. Smith, J. York, Amazon. com recommendations: Item-to-item collaborative filtering, Internet Computing, IEEE 7 (1) (2003) 76–80.

[22] R. M. Bell, Y. Koren, C. Volinsky, The bellkor solution to the netflix prize (2007).

[23] J. Vermorel, M. Mohri, Multi-armed bandit algorithms and empirical evaluation, in: Machine Learning: ECML 2005, Springer, 2005, pp. 437–448.

[24] P. Auer, Using confidence bounds for exploitation-exploration trade-offs, The Journal of Machine Learning Research 3 (2003) 397–422.

[25] S. Agrawal, N. Goyal, Thompson sampling for contextual bandits with linear payoffs, arXiv preprint arXiv:1209.3352.

[26] L. Li, O. Chapelle, Open problem: Regret bounds for thompson sampling., in: COLT, Citeseer, 2012, pp. 43–1.

[27] S. J. Russell, P. Norvig, Artificial Intelligence A Modern Approach, 3rd Edition, Prentice Hall series in artificial intelligence, Pearson Education, Boston, 2010.

[28] H. Polat, W. Du, Svd-based collaborative filtering with privacy, in: Proceedings of the 2005 ACM symposium on Applied computing, ACM, 2005, pp. 791–795.

[29] G. Shani, A. Gunawardana, Evaluating recommendation systems, in: Recommender systems handbook, Springer, 2011, pp. 257–297.

[30] A. Said, A. Bellogín, Comparative recommender system evaluation: benchmarking recommendation frameworks, in: Proceedings of the 8th ACM Conference on Recommender systems, ACM, 2014, pp. 129–136.

[31] O. Chapelle, L. Li, An empirical evaluation of thompson sampling, in: Advances in neural information processing systems, 2011, pp. 2249–2257.

[32] Y. Z. Wei, L. Moreau, N. R. Jennings, Learning users' interests by quality classification in market-based recommender systems, Knowledge and Data Engineering, IEEE Transactions on 17 (12) (2005) 1678–1688.

[33] The acm conference series on recommender systems (2015).
URL `http://recsys.acm.org/`

[34] Z.-D. Zhao, M.-S. Shang, User-based collaborative-filtering recommendation algorithms on hadoop, in: Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on, IEEE, 2010, pp. 478–481.

[35] J. Wang, A. P. De Vries, M. J. Reinders, A user-item relevance model for log-based collaborative filtering, in: Advances in Information Retrieval, Springer, 2006, pp. 37–48.

[36] J. Wang, A. P. De Vries, M. J. Reinders, Unifying user-based and item-based collaborative filtering approaches by similarity fusion, in: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2006, pp. 501–508.

[37] M. Deshpande, G. Karypis, Item-based top-n recommendation algorithms, ACM Transactions on Information Systems (TOIS) 22 (1) (2004) 143–177.

[38] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th international conference on World Wide Web, ACM, 2001, pp. 285–295.

[39] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer (8) (2009) 30–37.

[40] J. S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

[41] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, Springer, 1998.

[42] P. Kazienko, P. Kolodziejski, Personalized integration of recommendation methods for e-commerce., IJCSA 3 (3) (2006) 12–26.

[43] M. A. Ghazanfar, A. Prugel-Bennett, A scalable, accurate hybrid recommender system, in: Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on, IEEE, 2010, pp. 94–98.

[44] S. Trewin, Knowledge-based recommender systems, Encyclopedia of Library and Information Science: Volume 69-Supplement 32 (2000) 180.

[45] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, Knowledge and Data Engineering, IEEE Transactions on 17 (6) (2005) 734–749.

[46] J. G. March, Exploration and exploitation in organizational learning, Organization science 2 (1) (1991) 71–87.

[47] A. G. Barto, Reinforcement learning: An introduction, 1998.

[48] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Machine learning 47 (2-3) (2002) 235–256.

[49] W. R. Thompson, On the likelihood that one unknown probability exceeds another in view of the evidence of two samples, Biometrika (1933) 285–294.

[50] J. Wyatt, D. Bu, Exploration and inference in learning from reinforcement.

[51] S. Agrawal, N. Goyal, Analysis of thompson sampling for the multi-armed bandit problem, arXiv preprint arXiv:1111.1797.

[52] D. Evans, The internet of things how the next evolution of the internet is changing everything. cisco internet business solutions group, ibsg (2011).

[53] Y. Zhou, X. Zhang, X. Jiang, V. W. Freeh, Taming information-stealing smartphone applications (on android), in: Trust and Trustworthy Computing, Springer, 2011, pp. 93–107.

[54] G. Greenwald, E. MacAskill, Nsa prism program taps in to user data of apple, google and others, The Guardian 7 (6) (2013) 1–43.

[55] D. M. Berry, Internet research: privacy, ethics and alienation: an open source approach, Internet research 14 (4) (2004) 323–332.

[56] A. S. chiu, The ethics of internet privacy.

[57] D. Goldschlag, M. Reed, P. Syverson, Onion routing, Communications of the ACM 42 (2) (1999) 39–41.

[58] M. G. Reed, P. F. Syverson, D. M. Goldschlag, Anonymous connections and onion routing, Selected Areas in Communications, IEEE Journal on 16 (4) (1998) 482–494.

[59] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: ACM Sigmod Record, Vol. 29, ACM, 2000, pp. 439–450.

[60] S. Y. Jung, J.-H. Hong, T.-S. Kim, A statistical model for user preference, Knowledge and Data Engineering, IEEE Transactions on 17 (6) (2005) 834–843.

[61] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Analysis of recommendation algorithms for e-commerce, in: Proceedings of the 2nd ACM conference on Electronic commerce, ACM, 2000, pp. 158–167.

[62] A. Rajaraman, J. D. Ullman, Mining of massive datasets, Cambridge University Press, 2011.

[63] Facebook licence agreement (2015).
URL `https://www.facebook.com/about/privacy`

[64] E. L. Barse, H. Kvarnstrom, E. Jonsson, Synthesizing test data for fraud detection systems, in: Computer Security Applications Conference, 2003. Proceedings. 19th Annual, IEEE, 2003, pp. 384–394.

A

The Data Views

```
SELECT t2.Id AS "UserId",
        t1.DALI_Created AS "Updated",
        t1.ZipCode AS "ZipCode",
        t2.DateOfBirth AS "DateOfBirth",
        t2.Gender AS "Gender"
FROM dbo.UserAddresses t1 JOIN dbo.UserDetails t2 ON t1.UserId
   = t2.Id
WHERE ZipCode LIKE '[0-9][0-9]%' AND ZipCode NOT LIKE '%[,]%'
```

Listing A.1: Query to extract valid userdata.

```
SELECT t1.ProductId AS "ItemId",
        t2.Gender AS "Gender",
        t1.DALI_Created AS "ItemCreated",
        t1.DALI_Modified AS "ItemModified",
        t1.CurrentStock AS "CurrentStock",
        dbo.toBinaryString(t2.Category) AS "Category"
FROM ProductDetails t1 JOIN Products t2 ON t1.ProductId = t2.Id
```

Listing A.2: Query to extract valid itemdata.

```
SELECT t2.UserId AS "UserId",
       t2.Id AS "OrderId",
       t3.ProductId AS "ProductId",
       t2.DALI_Created AS "OrderCreated"
FROM (CustomerOrderItems t1 JOIN
       CustomerOrders t2 ON t1.CustomerOrderId = t2.Id) JOIN
       ProductDetails t3 ON t1.ProductDetailId = t3.Id
WHERE EXISTS (SELECT UserId
                          FROM UserData s1
                          WHERE t2.UserId = s1.UserId)
```

Listing A.3: Query to extract valid orderdata.