Constructing & Evaluating Context-Aware Recommender System in a case study with webshop carts and AB-testing

Master of Science Thesis in Algorithm, Language and Logic

Alexander Lundgren & Linus Lindberg

Department of Computer Science & Engineering
Chalmers University of Technology
Gothenburg, Sweden 2014

Constructing & Evaluating Context-Aware Recommender System in a case study with webshop carts and AB-testing

Alexander Lundgren
Linus Lindberg

Examiner: Devdatt Dubhashi

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

Department of Computer Science and Engineering
Göteborg, Sweden August 2014

Abstract

A potential customer spends around five minutes on a webshop and visits roughly ten pages. To maximise sales it is crucial that the most relevant products is presented within those limits. The solution is to use a recommender system that predicts and recommends items on a personal level to the customer using collaborative filtering. Therefore, the project aim was to construct a working prototype for 3Bits and Lindex with an AB-testing phase to validate performance. Such evaluation of the collaborative filtering paradigm is not very common and as it only focus on on-going carts with binary implicit feedback, it does not take other information sources into account.

To thoroughly evaluate the performance in this particular configuration five test-phases was used: experimental prototype cross-validation tests, prototype testing using historical data sets, prototype testing using contextual pre-filtering, prototype cross-validation of synthetic generated data and finally AB-testing. A lot of literature and data was studied in order to be able to construct the evaluation tests. Furthermore, to enable AB-testing 3Bits had to integrate the prototype into Lindex web page in a way that maintained the professional level.

The evaluation shows a high accuracy compared to recommendations based on the most frequent occurring items. Furthermore, results from AB-tests of the project-algorithm against the old recommendation services at Lindex showed contradicting results. Additionally, the evaluation also showed that incorporating contextual pre-filtering to the prototype did not increase performance.

Keywords: Recommender systems, Collaborative filtering, Singular Value Decomposition, K-means, Context, Concept drift, Distribution, Ranking.

Acknowledgements

Our master thesis could not have been realised without the help and support from 3Bits AB whom have been deeply involved with the project. An excellent and friendly company to work with! Chalmers University of Technology have our thanks for accepting and promoting the project. We also want to especially thank:

- Devdatt Dubhashi, our examiner and supervisor from Chalmers, whom brought reason and direction when we most needed it.
- Jonas Hörnstein, the supervisor from 3Bits, for excellent guidance and feedback.
- The personnel at 3bits, for very friendly support and patiently answering questions when ever needed.
- Lindex for giving us the opportunity to work with their data and allowing AB-testing to be performed.

A. Lundgren L. Lindberg, Gothenburg 2014-08-31

Contents

1 Introduction **1**

1.1 Background 2

1.2 Purpose 3

2 Theory **5**

2.1 Definitions & project structure 5

2.2 Generating feedback 7

2.3 Creating feature space 9

2.4 Clustering categories 15

2.5 Classifying an item 17

2.6 Finding similar items 18

2.7 Creating recommendation 19

2.8 Justifying choices 27

3 Methods & Materials **30**

3.1 Procedure 30

3.2 Materials 31

3.3 Participants in study 32

3.4 ItemRand- & frequent-algorithm 32

3.5 Test procedures 33

4 Result **41**

4.1 Basic tests 41

4.2 Time tests 43

4.3 Context tests 54

4.4 Synthetic testing 57

4.5 AB-tests 58

4.5.1 Product Page 58

4.5.2 Checkout Page 58

5 Discussion **60**

6 Conclusion **67**

Bibliography **69**

Appendices **72**

1

Introduction

An average visitor on an e-commerce site spends roughly five minutes and visits around ten pages in their visit. This makes it crucial to present the correct information to the customer in the period of time they are visiting to maximize the probability of a sale.

The conventional approach of navigation has been static structures of products and offers, usually in form of hierarchical menus or search functions which assumes the customer knows what s/he needs. However, there exists services that can foresee what products will attract the customer to a purchase. Such services belongs to a category called recommender system(RS) (Jannach, Zanker, Felfernig, & Friedrich, 2011). Their main objective is to recommend items the user unknowingly demands. These services is effectively presenting correct products faster than what a conventional navigation system would have done (Ricci, Rokach, Shapira, & Kantor, 2011).

On the other hand, some predictions may be inaccurate which will yield less purchases using such structures. This occur as the system models predict irrelevant products in the visiting period which does not attract the customer to a purchase. The RS would preferably be improved to yield even more accurate predictions and hence, better sale results.

Contextual information can be used in a recommender system to improve the prediction accuracy. Unfortunately, context has many definitions and for the curious reader there exist a conference who's main goal is to define context (Polytech, 2013). However, the general description of contextual information is information that fits a certain context, which might not be very explanatory. Another way of describing what context is could be the statement of Musto, Semeraro, and Lops (2013, p.130): *"a set of factors able to influence user perception of the utility of a certain item"*. Furthermore, traditional recommender system only have two entities, users and items (Adomavicius & Jannach, 2013, p. 1). In contrast, context-aware recommender system(CARS) uses contextual information such as time, location, weather, the users mood or type of device etc. This creates more entities connected to a user than simply just users and items, which serves as one of the underlying factors that could make CARS generate more accurate predictions.

1.1 Background

The field of recommender systems have been around a while and much research have been done, especially on the three major types of recommender systems: collaborative filtering, content-based and knowledge-based. Their main difference is the information they use to construct recommendations. Collaborative filtering uses similarities between users and items. Content-based uses static information about users or items. Such information can be descriptions where the text is evaluated to generate prediction values regarding the relevance of a particular item or user. Knowledge-based are dependent on information that are obtain directly from the users, for example from questions or quiz.

The information recommender systems handles are generally divided into two categories of relevance feedback: implicit feedback and explicit feedback. A more detailed description of these concepts is available in the theory section. However, information given by the user is said to be explicit and gives a high truth value but only for items the user chose to reveal her preference about. Furthermore Implicit feedback is gathered by analysing the user behaviour, such as patterns of clicks or number of items sold.

In this thesis the focus is going to be on implicit feedback as the available data does not contain user ratings and in general very little explicit feedback. To be too dependent on user ratings might be disadvantageous as time will be spend from both the user and the system to generate enough data. However, constructing good implicit rating can be a greater challenge as a large amount of information sources has to be considered.

Unfortunately, the data is usually sparse. To counteract such problems, matrix factorization models that utilizes a concept called principal component analysis can be used, later explained in the theory section 2. The winners of the Netflix challenge (Thompson, 2008) used a matrix factorization technique called singular value decomposition(SVD) together with k-nearest neighbors(kNN) (Koren, Bell, & Volinsky, 2009). Furthermore, SVD uses the PCA concept to reduce the dimensionality of the data. SVD and kNN also works well together as SVD estimates general structures and kNN localizes relationships (Koren, 2008). However, the main contribution is the notion of how temporal drift could be captured.

Gap

It seems to exist much literature on RS but not as much on CARS which is endorsed by Musto et al. (2013, p. 127), *"Even if there exists a very vast literature on RSs, the research about context-aware RSs(CARS) is relatively new".* This suggests that the CARS is of interest for research.

The research field of CARS can be divided into four categories:

1. Fundamental - i.e., understanding the notion of context and modeling context in recommender systems.
2. Algorithms - i.e., developing recommendation algorithms that can incorporate contextual information into recommender systems in advantageous ways.
3. Evaluation - i.e., in-depth evaluation of context-aware recommender systems performance, their benefits and limitations.
4. Engineering - i.e., designing general-purpose architectures, frameworks, and approaches to facilitate the development, implementation, deployment, and use of context-aware recommendation capabilities.

—-Adomavicius and Jannach (2013, p. 2)

However, according to Adomavicius and Jannach (2013, p. 3) , *"As compared to the "Algorithms" category, the other three aforementioned categories have been relatively underexplored"*, seems most of the research to have been done in the algorithm section. This suggests our objective would be to work on the evaluation and engineering aspects of CARS.

Problem Description

A potential customer have a finite and usually short period of time s/he spends on a particular web-shop. To increase sales, it is of interest to present as many relevant products toward the customer as possible in that time, as relevant products is those that have a high probability of being purchased by the customer.

However, customers preferences differs, some times a lot. Their preferences might also change over time, such as around Christmas etc. This suggests the challenge lies in constructing a recommendation system that can manage drifting customers preferences but also categorize customers in regard to their personalities.

The key concept needed to solve the problem is context-awareness and concept drifting. These concepts have been around for awhile, but there is no standardized way of solving the problem using these concept with the constraint to only consider items in an on going cart. The concept drifting tends to be unique toward the domain of the problem, in this case e-commerce of cloths and accessories.

As many of the algorithms is explained from previous research, the task will be to find an efficient way to implement them and evaluate performance.

1.2 Purpose

To construct a CF RS prototype and evaluate it with several tests using historical data, contextual information and if it passes these test will also AB-testing be performed in Lindex webshop.

Research Question

Given a user that puts an item into his/her shopping cart, is it possible to create an implicit-feedback based collaborative filtering recommender system from shopping carts history that is interchangeable with a explicit-feedback based CF RS? How does this affect the performance in Neighborhood and factor models? Can the system be enhanced with contextual information and system adaptiveness in such way that it will improve performance, perhaps out perform a non-CARS?

Scope

The effort will be most needed in the evaluation and interpretation of concept drift for this particular data context, as much research have already been done in the algorithmic and engineering aspects of CARS. To best utilize the twenty working weeks will the focus be on implicit-cart-based prediction and how context & adaptiveness can be incorporated to catch concept drifts in the data. The major achievement from this work will be performance testing of CARS, therefore is it crucial that other aspects of the project will take second place such that evaluation can be of primary concern. The project will be delimited to only consider on going carts, but the spectrum can be widen to include site click, viewing time etc. if time allows.

Thesis outline

Chapter two contains eight sections where the first explains the structure and defines variables of the six succeeding sections. The six succeeding sections will first go through the theory needed for the algorithms then explain the algorithms with descriptive pseudo code. Lastly a section that justify the design chooses done in the algorithms.

Chapter two contains a theory section and a justifying choices section. The theory section explains the major algorithms used in the project. To fully understand the discussions, it is important that the reader is well versed in the main theories. The justifying choices section explains the project algorithm and the different choices made.

Chapter three describes the methods and material used to carry out the project. If the reader wants to reconstruct the project, this should be possible except from using the Lindex data set. Furthermore, the tests used to evaluate the project algorithm is explained in this part of the report as well.

Chapter four contains the results and their corresponding explanation. Chapter five is the discussion of the results presented in chapter four. Chapter six contains the conclusion based on the discussion in chapter five.

2

Theory

This chapter is divided into eight sections. These sections aims to explain how the structure of the recommendation systems is built in individual components and how those components are connected. Each component have their own section that contains an detail explanation. The chapter ends with a section that wraps all component together and justifies the design choices taken.

2.1 Definitions & project structure

These are the major definitions needed to fully understand the algorithm in the following sections.

$feedback$	The implicit feedback of N unique customers and M unique items constitutes an $N \times M$ matrix. The matrix is binary and has the value 1 for each customer and item that has a relationship.
dim	dim is used in the construction of the feature space, Ω. The preferred number of dimension Algorithm 2 reduces the Ω to.
Σ	A diagonal matrix of singular values in an decreasing order, obtained from performing SVD on the $feedback$ matrix.
V	The right singular values obtained from performing SVD on the $feedback$ matrix.
Ω	The feature space.
L	A List of labels for each item, explaining which cluster the item belongs to.

k	The total number of clusters to be used in categorisation were chosen to have a size of 80% of the unique items in the training set.
C	A list of each cluster's centroid in feature space.
$distribution$	An ongoing cart can have many personalities/cluster given a set of items. The distribution of these clusters is contained in $distribution$.
$dist$	$dist$ is one probability with its corresponding cluster as $distribution$ is a set of probabilities with corresponding clusters.
$itemDistrubtionPairs$	A list of tuples containing a list of items from one cluster and its distribution of the carts in question.
$nrRecs$	The number of items to be recommended.
$recommendedItems$	The list of items that the recommender have generated. It has the length of $nrRecs$.
$prob$	The probability that an item from a specific cluster should be recommended given multiple items.
$probSum$	The aggregation of probabilities given a specific cluster.
$hits$	Numbers of times a cluster exists in $manyDistributions$.
$manyDistribution$	A concatenated list of all cluster distributions.
$newDistribution$	An aggregation of distributions done in algorithm 4 using $manydistributions$.
Time period	A specific period in time that usually defines one or more set(s).
Project Algorithm	The recommender algorithm explained in Justifying choices.
randomItem Algorithm	Randomly selects an item of all items and recommends it.
Frequency Algorithm	Select the most frequent occurring item and recommends it.

Table 2.1

The abstract structure of the prototype is visualized in figure 2.1 which contains five major component/algorithms. These algorithms is explained one by one with theory and pseudo-code. After the explanation of the different algorithm will a justification section follow, that justifies different design choices but also linking the different algorithms together in an descriptive way.

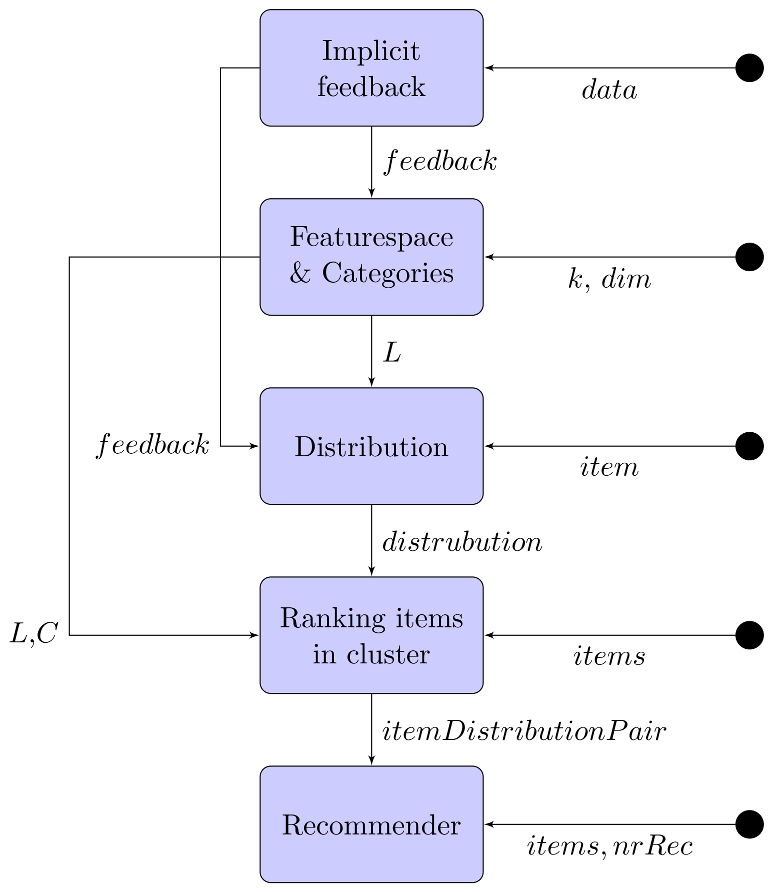


Figure 2.1: Flow chart visualizing the relationship between algorithms 1-6.

2.2 Generating feedback

This section will first introduce the reader to the theory behind the algorithm and end with a descriptive pseudo code.

Relevance Feedback

The underlying idea behind relevance feedback is simply: how relevant is a particular item in forthcoming queries? As an example, when recommending movies, how relevant is an action movie compared to a drama movie towards a particular user? This suggest the personality of users is captured by obtaining their preferences as feedback, either explicit or implicit.

An explicit relevance feedback, or just short: explicit feedback, is an active action from the user. An example of such action is ratings on movies, to follow the previous example structure, such as a ”five star”-structure or ”1-10”. The most obvious advantage for explicit feedback is its simplicity, no further calculations is needed. However, the disadvantage is that feedback will only be collected from what the users chooses to reveal to the system.

Implicit feedback collects a lot of information about the users such as site clicks, time spent on movie reviews etc. This data can then be transformed in to unintentional feedback or implicit feedback as the user has indirectly revealed her intention to the system. An advantage of this approach is that feedback can be obtained without the users explicit interaction and it might better capture the true personality of the user. It is also possible to capture preferences that the user might not even know she had herself, based on the pattern in the collected data. These pattern is usually referred to as unconscious pattern.

However, these unconscious pattern can only be obtain with the assumption that the user is simply one user and does not share account or in any other way have a schizophrenic personality.

Concept Drift

In the field of Machine Learning the term ”concept drift” usually refers to the phenomenon of changing statical properties from a target variable over time. The changes cannot always be foreseen and as can be imagine is somewhat cumbersome when trying to apply a predictive model. The trick is to alter the model in such a way that it will be able to manage the concept drift as well. Unfortunately, this is not easy as the changes can be unforeseen and needs prospective engineering.

Concept drift is usually perceived as characteristic towards the domain of the problem. One such concept drift of shopping pattern is occurring towards Christmas times where more money is spent than usual, and less than usual in January. The opposite concept drift can be observed regarding the time spent on working which suggests these two seemingly different activities might have an inverse correlation. However, to avoid being too philosophical, such perspective is not pursued. Even if most of the world is connected in one way or another, it might suffice to only consider basic preconception in e-commerce to capture the sought concept drift, all people spend more money around Christmas and less in January. Otherwise any attempt to actually compute a model that taking such massive concept drift into consideration, might turn out to be infeasible.

The Algorithm 1

Algorithm 1 generates binary implicit feedback from historical user purchase data. How the algorithm is connected to the system in general can be observed in the flow chart of figure 2.1. An justification of the algorithm and how its used will be done in the justifying choices section 2.8.

Algorithm 1: Implicit feedback

```
  Data: Customers and corresponding items bought
1 N = the number of unique customers in the dataset
2 M = the number of unique items in the dataset
3 $feedback$ = an N $\times$ M matrix
4 for $i \in N$ do
5     for $j \in M$ do
6         $feedback_{i,j} = 0$
7         if customer i have bought item j then
8             $feedback_{i,j} = 1$
9 return $feedback$
```

2.3 Creating feature space

This section will first introduce the relevant theory behind the algorithm and end with a descriptive pseudo code, as the previous section did.

Eigenvalue Decomposition

Let A be a quadratic matrix. If the scalar λ and column matrix X satisfy

$$AX = \lambda X \text{ where } X \neq 0 \tag{2.1}$$

X is said to be eigenvector and λ eigenvalue to A (Sparr, 1975, p. 238).

From equation (2.1) it is possible to generate the characteristic polynom for A, $p_A(\lambda) = det(\lambda I - A)$ (Sparr, 1975, p. 242).

$$p_A(\lambda) = det(\lambda I - A) = 0 \tag{2.2}$$

The eigenvectors with corresponding eigenvalues λ is composed from the solution to the characteristic equation described in equation (2.2).

Furthermore, let $e_1, \dots, e_n$ and $e_1^{'}, \dots, e_n^{'}$ be two bases in $\Re^n$ (Sparr, 1975, p. 185). If the linear projection F is described with $Y = AX$ with the base $e_1, \dots, e_n$ and $Y^{'} = A^{'}X^{'}$ with the base $e_1^{'}, \dots, e_n^{'}$ then:

$$A^{'} = S^{-1}AS \tag{2.3}$$

Where the matrix used for change of basis S consists of $e_1^{'}, \dots, e_n^{'}$ as column vectors (Sparr, 1975, p. 247).

And a linear projection F is said to be diagonalizable if there exists a base in which the projection matrix for F is an diagonal matrix (Sparr, 1975, p. 247):

$$D = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \tag{2.4}$$

From equation (2.3) and (2.4) it then follows that a matrix A can be said to be diagonalizable if there exists an invertible matrix S and an diagonal matrix D such that (Sparr, 1975, p. 247):

$$S^{-1}AS = D \tag{2.5}$$

And an $n \times n$ matrix A is diagonalizable if and only if there is n linear independent eigenvectors to A. In turn, it follows from equation (2.5) that the column vectors of S is eigenvectors to A and the diagonal elements in D is the eigenvalues that corresponds to the eigenvectors (Sparr, 1975, pp. 247-248).

The eigenvectors in S can be obtained by solving equation (2.2) as the observant reader probably already have grasped.

Covariance

Covariance is a measurement of how closely two variables varies relative to one another (J. S. Milton, 1986). If a variable X varies in the same direction they are said to be positively related. However, if they are moving in the opposite direction they are said to be negatively related. This suggest covariance capture the linear relationship between variables which can be very useful in many applications.

In some situations the covariance is zero, this can be the case when the variables are independent. Unfortunately, this can not be told by a covariance as the variables might just vary in such way they yield zero covariance.

$$Cov(X{,}Y) = E[(X - \mu_X)(Y - \mu_Y)] = E[XY] - E[X]E[Y] \tag{2.6}$$

Equation (2.6) is the definition of covariance where $E[X]$ is the expectation of X and μ_X denotes the mean of X. This equation does not give any measurement on the relation between variables, it simple state there is a relation. However, the degree of relation between variables, correlation, can be determined by computing the Pearson coefficient of correlation.

Principal Component Analysis

The fundamental idea of principal component analysis(PCA) (Shlens, 2005) is to preserve the most relevant information from a data set while reducing the dimension to a more manageable size. Some times can data be overwhelming to such degree that it cannot be used. Such data is regarded as noisy. That is, elements of the data is distorting the overall picture.

If the data would be reduced into a lower dimension, the distorting elements might disappear. This could potentially reveal concentrations that would previously not been recognized in a higher dimension. Figure 2.2 visualizes the concept. The top plot shows the original data in three dimension. The middle plot shows the same data projected into a two dimensional plane. The bottom plot is the same projection but with the variance drawn as red vectors from the centroid of the data set.

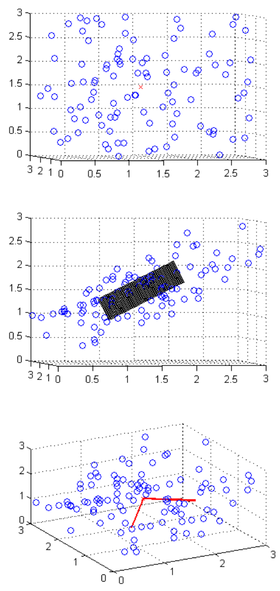


Figure 2.2: Dimension reduction, from 3 to 2 dimension

The issue is to figure out what dimensions preserves the most relevant information as it would be contra productive to reduce into dimensions that discard relevant information and amplifies distortion. Distortion or noise in data occurs as the correlation between elements becomes so intense they are barely distinguishable. Such premise implies that the opposite is pursued. That is, dimensions that obtains high variance of the set but little covariance. High variance of the set is preferable as elements divergence from the centroid is increased. If the data set were to be projected toward the vector of the variance, than less correlation would most likely be obtained as a result.

This insight suggest dimension reduction should consider variance vectors in an descending order, a ranking, to best preserve relevant information. By only considering some of the highest ranked variances, the data might be better represented as noisy dimension is discarded, this would be lower ranks. Furthermore, the variance and correlation is obtain from the covariance matrix where the diagonal is the variance and the remaining elements is the correlation. Zero correlation is pursued as is illustrated in the following equation:

$$cov(A) = \begin{bmatrix} E[(A_1-\mu_1)(A_1-\mu_1)] & & 0 \\ & \ddots & \\ 0 & & E[(A_n-\mu_n)(A_n-\mu_n)] \end{bmatrix} \tag{2.7}$$

The vectors of projection constructed from the variance will become the new basis, the principal component. It is possible to see the basis transformation as a rotation and a stretch as these vectors will become the new axis. This will work if the vectors are orthogonal to each other, therefore the second variance have to be orthogonal to the first.

$$Y = PX \tag{2.8}$$

$$cov(Y) = C_Y = \frac{1}{n}YY^T \tag{2.9}$$

These new vectors is used to transform our data set X into Y by doing a rotation and a stretch with P as in equation (2.8). P has to be constructed in such way that equation (2.9) satisfy the condition in equation (2.7) , decorrelation. The row of P will then be the principal component of the data set X. Another way of putting it: C_Y is an diagonal matrix D where its diagonal elements is the variance. This matrix can be obtained by doing eigenvalue decomposition and selecting P as the corresponding matrix where each row is a eigenvector, $P \equiv E^T$.

$$C_Y = \frac{1}{n}(PX)(PX)^T = P(\frac{1}{n}XX^T)P^T = PC_XP^T \tag{2.10}$$

$$C_Y = P(E^TDE)P^T = PP^TDPP^T = IDI = D \tag{2.11}$$

C_Y can be reformulated to reveal its dependence on the covariance for the data set X, C_X, as is illustrated in equation (2.10). It is also possible to perform eigenvalue decomposition to obtain $C_X = EDE^T$ as the covariance of X is a square matrix. Furthermore, the columns of E are orthogonal vectors as C_X is a symmetric matrix, satisfying the constraint on orthonormal basis in order to construct the new euclidean space. In turn, satisfying decorrelation of C_Y. This allows equation (2.11) to be derived by using the eigenvalue decomposition theorem and selecting $P \equiv E$. The previous equation derives $C_Y = D$ where D is a diagonal matrix of eigenvalues corresponding to the eigenvectors in E.

To summarize: The problem can be reformulated as finding the principal component to change the basis of the euclidean space such that correlation is minimized assuming that the problem is of linear nature. This is accomplished by recognizing that large variances usually yield less correlation where the variance is the eigenvalues derived from eigenvalue decomposition. The corresponding eigenvectors is the principal component assuming the eigenvectors is orthogonal, which it is if the covariance of the data set is symmetric, satisfying the eigenvalue decomposition theorem.

Singular Value Decomposition

Singular Value Decomposition(SVD) (Shlens, 2005; Kalman, 1996; Austin, 2009) is a technique to factorize matrices. To generate many matrices out of one matrix such that their product will yield the original matrix. Such factorization is accomplished by first constructing the eigenvectors from the product between the original matrix, A, and its conjugate transpose, $U = A\overline{A^T}$. These eigenvectors is called the left singular values. The eigenvector of the conjugate transpose times the original matrix will yield the second factor called right singular values, $V = \overline{A^T}A$. The last factor, Σ, is obtain by taking the square root of the none zero Eigenvalues of both U and V. The matrix obtained looks like this:

$$A_{m,n} = U_{n,n} \cdot \Sigma_{n,m} \cdot V_{m,m} \qquad \text{s.t} \quad a \in A, u \in U, s \in \Sigma, v \in V \tag{2.12}$$

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix} = \begin{bmatrix} u_{1,1} & \cdots & u_{1,n} \\ \vdots & \ddots & \vdots \\ u_{n,1} & \cdots & u_{n,n} \end{bmatrix} \begin{bmatrix} s_{1,1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & s_{n,m} \end{bmatrix} \begin{bmatrix} v_{1,1} & \cdots & v_{1,m} \\ \vdots & \ddots & \vdots \\ v_{m,1} & \cdots & v_{m,m} \end{bmatrix} \tag{2.13}$$

These factors can be manipulated to generate approximation of the original matrix M by only taking a sub set of Σ in to account. That is, reduce the original dimension of Σ into $k \times k$ such that U and V is enforced to be reduced into $U : n \times k$ and $V : k \times m$. The resulting matrices is visualized as follows:

$$A_{m,n} \approx U_{n,k} \cdot \Sigma_{k,k} \cdot V_{k,m} \qquad \text{s.t} \quad a \in A, u \in U, s \in \Sigma, v \in V \tag{2.14}$$

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \vdots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix} = \begin{bmatrix} u_{1,1} & \cdots & u_{1,k} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ u_{n,1} & \cdots & u_{n,k} \end{bmatrix} \begin{bmatrix} s_{1,1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & s_{k,k} \end{bmatrix} \begin{bmatrix} v_{1,1} & \cdots & \cdots & v_{1,m} \\ \vdots & \ddots & \ddots & \vdots \\ v_{k,1} & \cdots & \cdots & v_{k,m} \end{bmatrix} \tag{2.15}$$

The product of the reduced matrix factors will usually yield a very good approximation with just a few singular values. This is possible as the singular values is the corresponding principal component in PCA which explains the intuition behind the algebra. Such approximation saves memory that is needed to store the matrix or image, depending on the context, by simply just retain the reduced matrices from SVD and discard the original matrix. It also reduces sparsity from the original matrix which is desired in recommender system construction.

Nonnegative Matrix Factorization

None-negative Matrix Factorization(NMF) factorizes its target matrix A into two factor matrices, W and H, such that their product will produce a close approximation of A, $A \approx WH$. However, SVD does recreate the full matrix A, but yields negative values. In-contrast, NMF generates only non-negative values as it is bound by such constraints.

$$F = \frac{1}{2}(A - WH)^2 \tag{2.16}$$

The measurement of approximation accuracy is expressed by a cost function F, for example euclidean distance function described in equation (2.16). A full recreation of the matrix can only be obtained if F equals zero otherwise some information has been lost.

$$L = min(\frac{1}{2}(A - WH)^2) \qquad \text{s.t } 0 \leq W, 0 \leq H \tag{2.17}$$

$$\begin{aligned}\frac{\partial L}{\partial W} &= (A - WH)H^T = 0 \\ \frac{\partial L}{\partial H} &= W^T(A - WH) = 0\end{aligned} \tag{2.18}$$

The intuition behind the factor generation is to simply minimize the cost function such that both factors does not break the none-negativity constraints, expressed in equation 2.17. Thus, finding the factors that satisfies the constraints is an optimization problem. One of the first solution that comes to mind is gradient descent. Unfortunately, such solution can be computational expensive; however, there exists more efficient solution such as multiplicative updates, suggested by Lee and Seung (2001). Unfortunately both variables will not be convex at the same time, giving little hope of actually finding a global minimum (Lee & Seung, 2001, p.3). Therefore will any solution most likely generate an approximation and not find an exact solution.

$$\frac{AH^T}{WHH^T} = 1 \tag{2.19}$$

$$W_{t+1} = W_t * \frac{AH^T}{W_tHH^T + \varepsilon} \tag{2.20}$$

Equation (2.19) is derived from the derivative of L, described by equation (2.18). Additionally, the update function, described by equation (2.20), is derived from equation (2.19). To avoid division by zero and guarantee a decrease in each iteration, ε is added to its denominator.

$$\begin{aligned}\frac{\partial L}{\partial H} &= AH^T - WHH^T = -0.1 \\ &\rightarrow \frac{AH^T}{WHH^T} = 1 - \frac{0.1}{WHH^T} \\ &\rightarrow \frac{AH^T + 0.1}{WHH^T} = 1\end{aligned} \tag{2.21}$$

When the derivatives of L equals one, a minimum has been found and $W_{t+1} = W_t$. However, if there still is a gradient, as is described by equation (2.21), a new W will be forged by the update function. This explains how the multiplicative update rule of NMF will eventually converge around a minimum.

Moreover, there exists plenty of NMF variations which is categorized into four genres:

1. *Basic NMF (BNMF), which only imposes the non-negativity constraint.*
2. *Constrained NMF (CNMF), which imposes some additional constraints as regularization.*
3. *Structured NMF (SNMF), which modifies the standard factorization formulations.*
4. *Generalized NMF (GNMF), which breaks through the conventional data types or factorization modes in a broad sense.*

—(Wang & Zhang, 2013, s. 1337)

By looking at the categorisation definition, the example function in equation 2.17 is a Basic NMF. However, there exists different cost function which is referred to as sub-categories or types of BNMF. Furthermore, the initialisation of the matrix factor directly affects the outputs. If an random initialization occurs, an close approximation could be made in the first runs or the initialization could be way off and consume a lot of time in order to obtain the desired output.

CNMF adds more constraint to H and W such as regularization terms. Due to such prior knowledge will the constraints help find a solution faster. Furthermore, the cost function for the CNMF's is described by the following equation from Wang and Zhang (2013, s. 1344):

$$F = F + \alpha J_1(W) + \beta J_2(H) \tag{2.22}$$

The variables J are penalty terms to enforce constraints. The regularization terms α and β is used to balance the constraints such that over-fitting is minimized.

The algorithm 2

Algorithm 2 constructs the feature space given the implicit feedback from algorithm 1. There are other variables needed as well such as *dim* which is used to construct the feature space, Ω, by simply taking the first *dim* rows of Σ and V which would correspond to the principal components. That is, *dim* is the preferred number of dimension algorithm 2 reduces the feature space to. Other important variables is L and C which is explained in the definition table in the beginning of this chapter 2.1.

Algorithm 2: Featurespace & Categories

```
  Data: Feedback matrix $feedback$, Dimensions $dim$, Number of clusters $k$
1 [U, Σ, V] ← Perform singular value decomposition on $feedback$
2 $\Sigma^{'}$ ← Reduce Σ to the dimension dim
3 $V^{'}$ ← Reduce V to the dimension dim
4 $\Omega = \sqrt{\Sigma'} * V'$
5 $[L, C]$ ← Perform k-mean with $k$ clusters on Ω
6 return $L$ & $C$
```

2.4 Clustering categories

This section presents the relevant theory to give an understanding of clustering and categories.

Similarities

Each of the two factors generated from a matrix factorization algorithms, previously explained, can be used as a feature-space. All the remaining principal component is regarded as a feature or a topic, a latent relation. It is unknown what relation is represented by the features, but they are used as the new basis.

The feature-space, V or U, is a Cartesian coordinate system of n dimensions or features, $\Re^n$. Each dimension/feature of a item/term expresses the level of appurtenant towards a feature/topic. Enabling the measurement of similarity between terms. However, there exists different approaches to compute the similarity. The two most common approaches is euclidean distance and cosine.

$$V = \begin{bmatrix} \boxed{0.4} & 0.2 & \boxed{0.7} & 0.3 & 0.8 \\ \boxed{0.6} & 0.5 & \boxed{0.3} & 0.5 & 0.3 \\ \boxed{0.1} & 0.3 & \boxed{0.4} & 0.6 & 0.3 \end{bmatrix}, A = \{0.4, 0.6, 0.1\}, B = \{0.7, 0.3, 0.4\} \tag{2.23}$$

Equation (2.23) describes how two points in the feature-space, $\Re^3$, cohere. The columns can then be interpreted as vectors in $\Re^n$ where cosine and euclidean distance express their correlation between topics.

The most frequently used similarity is cosine in the field of recommendation systems. However, the choice of similarities is dependent on the particular data set composition. In some cases does other similarities yield better result, and Qian, Sural, Gu, and Pramanik (2004) states that similarities in higher dimensional data performs alike.

Another paper (Lathia, Hailes, & Capra, 2008) has evaluated even more similarity measures and has concluded that in the general case of collaborative filtering is the accuracy not effected by what similarity measure is being used.

To illustrate the concept better, figure 2.3 shows how the vectors A and B looks like in the feature-space. Vectors with high correlation will have a small euclidean distance or degrees in angle between each other and thus have a higher probability to form a cluster.

As an example, A could be the term "tissue" and B could be "cells", both belongs to the cluster that contains terms of medicine. The topics produced by the matrix factorization models could have found this relation already but it is more likely to find several less obvious relations that together form clusters that would be the true topic, in this case medicine. The topics, modelled as the Cartesian axis, could be cancer as z and cell research as x. However, A and B would have less relation with eyes that might have been the y axis.

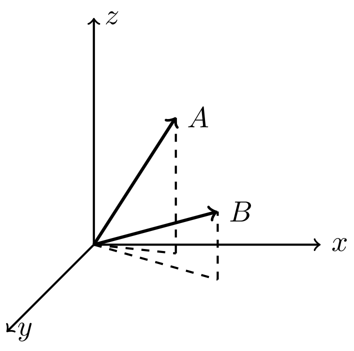


Figure 2.3: A & B from equation (2.23) is illustrated as vectors in feature space

K-mean

The K-mean algorithm[1] attempts to minimize the mean of the squared euclidean distance between points in an n-dimensional space. Intuitively, the algorithm tries to find a set of

[1]`http://en.wikipedia.org/wiki/K-means_clustering`

means that are as close as possible to all points. To accomplish this, two steps is taken for every iteration t.

$$s_i^t = \{x_p : ||x_p - m_i^t||^2 < ||x_p - m_j^t||^2 \quad \forall j, 1 \leq j \leq k\} \tag{2.24}$$

$$m_i^{t+1} = \frac{1}{|s_i^t|} \sum x_{j_{x_j \in S_i^t}} \tag{2.25}$$

Equation (2.24), which is the first step, defines the set of points belonging to every mean. A point can only be apart of a particular mean:s set if there are no other mean that have a smaller euclidean distance toward that point, as is described mathematically in equation (2.24).

The goal of the algorithm is to obtain a set of means with an optimal distance toward all points, which is visualized in table 2.2. That is, a set of mean with a minimal euclidean distance toward all points. However, as the means is usually randomly initiated, their initial state is seldom the optimal and it makes sense to update their position in order to minimize the distance toward the points further.

The second step of the algorithm, equation (2.25), actually tries to update the mean to better fit the data and satisfying the goal. By stepping toward the new mean, that has been computed from the set of points defined by the previous iteration:s mean, the euclidean distance is reduced by a smaller amount as is also described mathematically in equation (2.25).

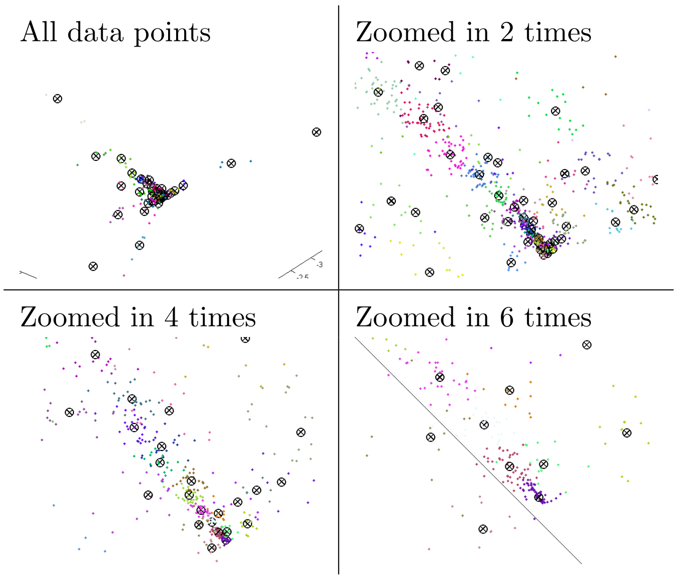


Table 2.2: Visualizes how k-mean clusters the data points in three dimensions using Matlab tools

However, it is not unusual for points to change their appurtenant as the mean:s position change. If too many points has to change appurtenant, which happens when the mean:s have to big step sizes, the algorithm convergence rate will be decreased, but that will also occur with a to small step size. Moreover, a global optimum might not be possible to find, depending on the initialization.

2.5 Classifying an item

The following algorithm computes the distribution of cart personalities given an item as explained previously. Algorithm 3 takes L from algorithm 2 as input in order to differentiate between the clusters. The variables are defined in table 2.1.

Algorithm 3: Distribution

```
Data: feedback $A$, Item $item$, Cluster Label $L$
1 $similarCustomers$ ← Find customers that have bought $item$ in $A$
2 for Every customer $c$ in $similarCustomers$ do
3     $carts$ ← The carts of customer $c$
4     for Every $cart$ in $carts$ do
5         if $cart$ contains $item$ then
6             for every item $cItem$ in $cart$ do
7                 $cLabel$ ← lookup cluster appurtenant for $cItem$ in $L$
8                 $distribution[cLabel]$ += 1
9 return Normalized $distribution$
```

Algorithm 3 constructs a distribution given one items. However, an ongoing cart can contain multiple items, and in these situations is the algorithm enhanced such that for every item is a distribution fetched and then concatenated into a new single distribution. Algorithm 4 does exactly that, it concatenates all relevant distributions and returns their aggregation. To aggregate the distributions is weighting used such that the probabilities for items occurring multiple times over many distributions have a higher weight in the new distribution.

Algorithm 4: Many Distribution

```
Data: $manyDistribution$
1 for Every cluster $c$ do
2     for Every distribution of $manyDistribution$ do
3         $hits$ ← Count the number of occurrence of cluster $c$
4         $probSum$ ← Aggregates the recommendation probability given cluster $c$
5     $prob$ ← $2^{hits-1} * probSum$
6     $newDistribution$ ← a pair of $c$ and the probability, $prob$, of recommending
      from $c$
7 return Normalized $newDistribution$
```

2.6 Finding similar items

As algorithm 3 computes the distribution of an ongoing cart does algorithm 5 compute which items should be recommended from these clusters. To do so is the clusters with the greatest probability considered, contained in *dist*. Then is the items in that particular cluster ranked from their euclidean distance towards the centroid of the cluster such that those items closes to the centroid is recommended first.

Algorithm 5: Ranking items in cluster

```
Data: List of customer selected items: $items$, Centroids $C$, Cluster labels $L$
1  $distribution$ ← Perform algorithm 3 given $items$
2  if No distribution generated then // Cold-start problem
3      return List of most frequent items
4  for Every dist in distribution do
5      $distItems$ ← Extract items belonging to $dist$ from $L$
6      for every item in distItems do
7          $c$ ← Find centroid in $C$, corresponding to $dist$
8          $eDistance$ ← Compute euclidean distance between $item$ and $c$
9      $itemList$ ← Sort the $eDistance$:s in descending order, switch to its items
10     $pair$ ← Construct a tuple of $dist$ and $itemList$
11 return A list of all $itemDistrubtionPairs$
```

Algorithm 5 is necessary to determined which item to select from a cluster such that the least representative item for the cluster is *not* selected.

2.7 Creating recommendation

This chapter will explain how to create recommendations based on carts. In the beginning is the theory explained and in the end is the algorithm expressed in pseudo-code.

Recommender Systems

This section will go through the major paradigms of RS in more detail to establish a better understanding for the reader. A good understanding will be of importance when discussing why the paradigm called collaborative filtering will be used.

Content-based recommender systems

Using content such as technical descriptions to make recommendations is usually easier than gather relevance feedback, assuming content exists. Recommendations is constructed by finding similarities between items content given the user preference. As Jannach et al. (2011, p. 54) has put it: "*To make recommendations, content-based systems typically work by evaluating how strongly a not-yet-seen item is "similar" to items the active user has liked in the past*".

Content-based RS mainly process item/user information in three components. First, the content analysis component where information is processed to be relevant and structured, an item representation of its content. A typical example is the transformation of a document into a set of keywords. Second, the profile learner component. It uses the output from the content analysis and user feedback to generate user profiles. The feedback could be user browsing or the like/unlike information. The profiles are usually built by machine learning techniques that generates models for each user, based on the feedback. Last, the filtering component where the user profiles are exposed to relevant items to generating a list of items to be recommended (Ricci et al., 2011, p. 75-76).

The advantage is the absence of user interaction required to construct ratings. Relevance feedback sparsity is of no concern as an additional advantage. However, if content information is not supplied then data sparsity is still a problem. The possibility of such situations is an disadvantage. So is misguiding information such that only certain aspects of the content is captured by the system (Ricci et al., 2011, p. 78), leading to erroneous recommendations. Another disadvantage is the risk of overspecialisation. An example of overspecialisation is the recommendation of more t-shirts as the user have been browsing t-skirts. It is not wrong in itself but it's preferable to recommend other categories that goes well with t-skirts as a users t-skirt demand will rapidly become saturated.

Knowledge-based recommender systems

There are situations where purchases from the same user is so rare that the user feedback will be out of date. In these situations will content-based and collaborative filtering paradigms do no good. However, there is a remedy: the knowledge-based paradigm.

There are mainly two types of this paradigm: case-based and constraint-based(Jannach et al., 2011, p. 82). Both uses requirements obtained from the users, then tries to construct recommendations based on these requirements. The main difference between the types is how their requirements differentiate. The case-based type tries to retrieve similar items from similarity measurements while the constraint-based type uses a set of explicit defined rules in order to construct recommendations.

This paradigm is preferably used in a more expensive context such as sales of cameras, cars, houses etc. These items does not usually generate any quantity of purchases and therefore produce to sparse data for other paradigms. However, the disadvantage is acquisition of the knowledge (Adomavicius & Tuzhilin, 2005, p. 23), it is explicit and users might dislike the enforced interaction.

Collaborative filtering

Collaborative filtering is the paradigm that will be used in this project as has been mentioned many times before. This chapter aims to explain this paradigm in detail with both general theory and examples. However, the underlying techniques has been explain previously in this section and will not be further explained here. The motivation and composition of theories will be done in the section Justifying choices.

The paradigm takes a matrix of user-item relations as argument then the following procedure is executed:

a) *A (numerical) prediction indicating to what degree the current user will like or dislike a certain item.*

b) *A list of n recommended items.*

— Jannach et al. (2011, p. 13)

The matrix are built up from a set of users $U \in \{u_1,u_2...,u_i\}$ and a set of products/items $P \in \{p_1,p_2...,u_j\}$. The cells of the corresponding $U \times P$ matrix consists of ratings $r_{i,j}$ for all products for each user(Jannach et al., 2011, p. 13). However, if the user have not rated the product its cell will be 0. This adds up to the following matrix:

$$A = \begin{array}{c} \\ u_1 \\ u_2 \\ \vdots \\ u_i \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & \cdots & p_j \end{array} \\ \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,j} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ r_{i,1} & r_{i,2} & \cdots & r_{i,j} \end{pmatrix} \end{array} \tag{2.26}$$

The rating in the matrix is either a gap, represented by a zero, or rated in some scale such as the "1-5 stars" or 1-10. The goal of the recommender is to present the user with the top n preferable products that s/he have not yet purchased or seen. In other words the gaps of the current user. To fill in the gaps and be able to recommend among those products, the paradigm finds similarity's between the different users based on historical data. There are many such similarity measures so lets take a look at a simple one just to prove the general idea of collaborative filtering. The example measure is the Pearson's correlations coefficient. Here's the example rating matrix:

$$A = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{pmatrix} 1 & 4 & 3 & 0 \\ 2 & 2 & 3 & 5 \\ 3 & 2 & 2 & 5 \\ 2 & 2 & 1 & 2 \end{pmatrix} \end{array} \tag{2.27}$$

As can be observed in equation 2.27, a gap exists for product p_4 at user u_1 as $r_{1,4} = 0$. This occurs as u_1 have not had any interaction with product p_4. Therefore, the system have to compute artificial ratings to fill in the gap. This task is accomplished by predicting the rating user u_1 would have for product p_4 by looking at similar users and hope they have a preference about the product that can be applied for the current user.

To make the example more simple will only one value be equalised zero. Unfortunately, there is a lot of gaps in real world data and this issue is usually referred to as sparsity, as has been mentioned previously.

To fill in the gaps, the collaborative filtering technique used in this example computes the correlation between users. In this example is the similarity's between users constructed with Pearson's correlations coefficient, a similarity measure between user-user, from the following formula(Jannach et al., 2011, p. 14):

$$sim(a,b) = \frac{\sum_{p\in P}(r_{a,p} - \bar{r_a})(r_{b,p} - \bar{r_b})}{\sqrt{\sum_{p\in P}(r_{a,p} - \bar{r_a})^2}\sqrt{\sum_{p\in P}(r_{b,p} - \bar{r_b})^2}} \tag{2.28}$$

In the $sim(a,b)$, a and b are users. $r_{a,p}$ and $r_{b,p}$ are ratings of item p for user a resp b. $\bar{r_a}$ and $\bar{r_b}$ is the average rating for user a resp b. By applying equation (2.28) on to matrix (2.27), the following matrix can be constructed:

$$A = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{array} \begin{array}{c} \begin{array}{cccc} u_1 & u_2 & u_3 & u_4 \end{array} \\ \begin{pmatrix} 1 & -0.8165 & -0.4264 & 0.8333 \\ -0.8165 & 1 & 0.5222 & -0.8165 \\ -0.4264 & 0.5222 & 1 & 0 \\ 0.8333 & -0.8165 & 0 & 1 \end{pmatrix} \end{array} \tag{2.29}$$

Each cell represent how similar the two users are as positive value represents positive correlation and negative values represents negative correlations. However, a value of zero means it does not exist a relation of linear nature(J. S. Milton, 1986, p. 170). As the correlation matrix has been generated the remaining task is to finding the nearest neighbours. It is simply done by taking the user with highest correlation to the current user, enabling the construction of a rating function:

$$predRating(a,p) = \bar{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \bar{r_b})}{\sum_{b \in N} |sim(a,b)|} \tag{2.30}$$

In equation 2.30 is N the neighbourhood of users with high correlation to the current user, in this case it is $N = \{u_2, u_4\}$. Equation (2.31) is the calculations of equation (2.30) with numbers:

$$\begin{aligned} predRating(u1,p4) &= 2.667 + \frac{0.8333 * (2 - 1.75) + -0.8165 * (5 - 3)}{0.8333 + 0.8165} \\ &= 2.667 - 0.4255 \\ &= 2.2415 \end{aligned} \tag{2.31}$$

The gap of rating $r_{1,4}$ in equation (2.27) can now be filled with the result from equation (2.31), 2.2415. It is now possible to recommend this product based on the new rating. This is the general idea of the collaborative filtering paradigm.

Collaborative filtering are used in a wast variety of applications such as e-commerce systems, movie sites, news sites, advertising systems and more. However, the more successful ones are Netflix and AmazonG Linden (2003) which both uses collaborative filtering to construct recommendations.

The primary advantage is scalability and that it fits well in many applications. But there are some disadvantages to collaborative filtering, the most critical one is the lack of knowledge/information usage compared to content-based and knowledge-based approaches.

Hybrid recommender systems

It would be profitable to combined the different paradigms to cancel out their weaknesses. The realisation of such notion is called hybrid recommender system. It is categorized into three general designs to merge the different paradigms (Jannach et al., 2011, pp. 124-139). The first design is the Monolithic design, it takes two or more paradigms and combine them such that their input is the same, the best results is chosen to be the output. The second design is the parallelized design, it takes multiple paradigms and aggregate their results into one output. The design operates in three patterns: weighted, mixed & switching. The last design, pip-lined design, simply execute each paradigm after the other such that the inputs and outputs has to be design to fit the other recommenders systems. There are also different patterns of this design such as meta-level and cascade etc.

Context-aware recommender systems

The three primary paradigms has been presented and hopefully some what pedagogically explained. Their focus is to recommend the most relevant items to users but without consider contextual information such as time, weather, location, mood etc. Recommendation accuracy could be improved by considering such contextual information. However, before going down to the technical details lets discuss what context really means.

First of all there are no fundamental definition of context; still, many fields are using the term and many definitions exists. Fortunately, there is a conference called CONTEXT that attend this issue and has a great variety of research fields affiliated to it:

- *Computer Science*
- *Artificial Intelligence and Ubiquitous Computing*
- *Cognitive Science*
- *Linguistics*
- *Organizational Sciences*
- *Philosophy*
- *Psychology*
- *Application areas such as Medicine, Law, domotics, context-aware systems, ...*

— Polytech (2013)

The huge variety of research fields indicates the difficulties in constructing a definition that could be applied to any discipline. Unfortunately, when Bazire and Brézillon (2005) tried to cover the issue by considering 150 definitions of context, their conclusions was that no such definition could be found, but a model of definitions. This is no surprising as some of the fields diverge a lot. Leading to as many definitions as there is research fields and researchers. In order to give the reader some insights about different views of context, the closely related fields Data mining, Information retrieval, E-commerce Personalization and Databases will be examined.

In Data mining is context described by Ricci et al. (2011, p. 220) as "*context is sometimes defined as those events which characterize the life stages of a customer and that can determine a change in his/her preferences, status, and value for a company*". In contrast, the field of E-commerce Personalization sometimes describes context as the intent of what the customer have for the purchase, gift, for children for them self's or some other intent (Ricci et al., 2011). The typical information used in recommender systems is user identification, item identification and what ratings each user has of a given item. The context information can be seen as some not typical information that is to be added into the recommendation phase to make better predictions.

Most of the implementations in information retrieval do not use contextual information, they simply use queries and collections to perform retrieval. However there exists some implementation that consider contextual information and they usually focus on web searching and immediate user interactions, for fast recommendations (Ricci et al., 2011, p. 222).

The last field is databases that uses context when returning queries. Some SQL-dialects have integrated ranking of the user precedences when the data is inserted into the database.

To get a general intuition of where in the recommendation process the contextual information is added the following example is presented: A matrix of ratings is used to construct recommendations as in previous sections such that rating gaps needs to be filled out. Fortunately, Ricci et al. (2011, p. 223) is able to do so with the function:

$$R : User \times Item \rightarrow Ratings \tag{2.32}$$

Equation (2.32) construct ratings to fill the gaps based on users and items only. However, when contextual information is incorporated, such that the system becomes context-aware, the equation is changed by Ricci et al. (2011, p. 224) in to:

$$R : User \times Item \times Context \rightarrow Ratings \tag{2.33}$$

There are two general approaches when constructing ratings using contextual information:

- *Context-driven querying and search*
- *Contextual preference elicitation and estimation*

— Ricci et al. (2011, p. 224)

The first approach uses contextual information directly in the query/search when going through the data storage, presenting the best matching result from the query/search. The second approach tries to model the user preferences by adding the contextual information to the model which will return user preferences that might not have been discovered otherwise. To manage this the recommender system paradigms can be extended or use machine learning techniques to learn the preferences.

In the context-aware recommender systems there are three major paradigms called pre-filtering, post-filtering and model-based context-aware paradigm. The first two are the opposite of each other so lets start with them. The pre-filtering context-aware paradigm uses the context to filter the input while post-filtering uses the context to filter the output. The third model-based paradigm adds the contextual information into the process of creating the model, most commonly by adding another dimension.

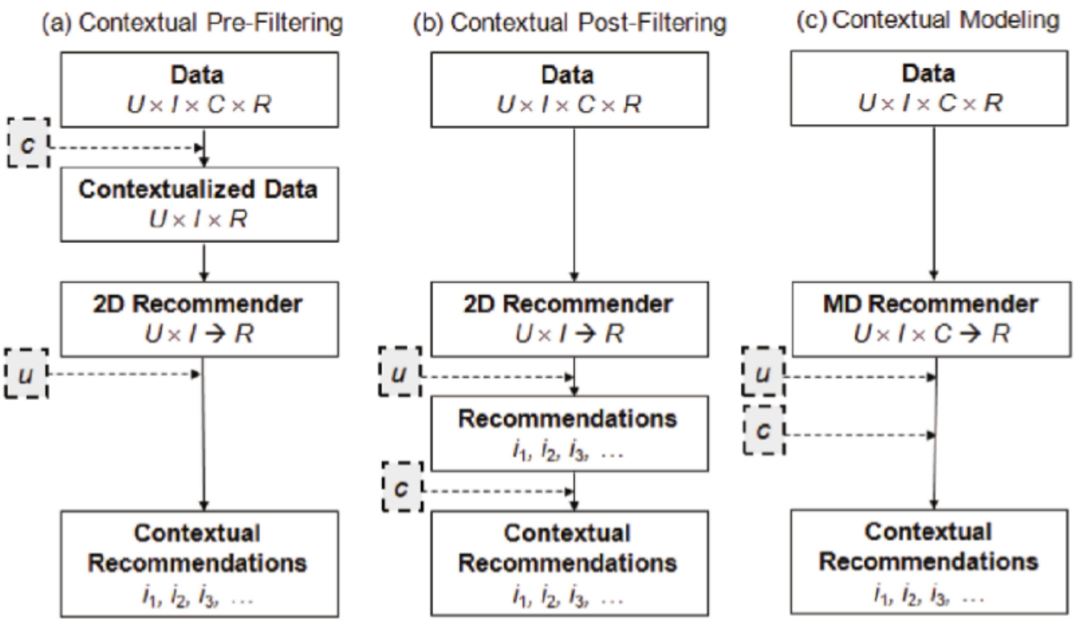


Figure 2.4: The figure depicts three context aware paradigms

The variables in figure 2.4 (Borrowed from Ricci et al. (2011, p. 233)) have the following definition:

- U: Set of users.
- I: Set of items.
- R: Set of ratings.
- C: Set of contextual information.
- i_1,i_2,...,i_n: recommended items in order of which to recommend.

There are three flow chart in figure 2.4, each of them depicts one of the three paradigms described above. Flow a) shows the pre-filtering where c explains where the context is added. Flow b) is the inverted flow of a) such that the context input c has moved to be added after the recommendation. Flow c) is the model-based approach where context is managed both inside the model and outside.

The Algorithm 6

This is the recommender algorithm used in this project that combining all the techniques to create recommendations. Table 2.1 visualizes how the different techniques is linked together. However, an important remark on algorithm 6: If not enough items is generated than the most frequent items will fill out the empty recommendations left. This is not explained in the pseudo-code but is still important to mention as it affects results.

Algorithm 6: Recommender

```
Data: Database, Customer selected items: $items$, Number of recommendation
      acquired: $nrRecs$
1  $feedback \leftarrow$ Algorithm 1 given data from database
2  $dim = 10$
3  $k = 80\%$ of unique items size
4  $[L, C] \leftarrow$ Algorithm 2 given $feedback$,$dim$ & $k$
5  $distributions \leftarrow$ Algorithm  3 given $feedback$, $items$ & $L$
6  $pList \leftarrow$ Algorithm 5 given $items$, $C$ & $L$
7  for each dist in distribution do
       // distribution is descending, highest probability first
       // a pair consists of the given distribution probability with
       // the corresponding cluster label
8      $pair \leftarrow$ Select a pair in $pList$, corresponding to $dist$
9      while There is items in pair do
10         $item \leftarrow$ first item of $pair$
11         Add $item$ to $recommendedItems$
12         Remove $item$ from $pair$
13         if size of $recommendedItems$ == $nrRecs$ then
14             return $recommendedItems$
15 return $recommendedItems$
```

2.8 Justifying choices

The construction of a recommender system is nothing new, it has been done before. Thus, a vast abundance of literature and conferences exists with the focus of exploring and amplifying the technology. A consequence of such research wealth is the effort needed to go through all information, to create a foundation of theory that will increase the probability of making the right decisions. Furthermore, the preliminary literature study revealed three major paradigms (see section 2.7). As it is of major concern for performance to choose the correct theory, in this case a paradigm depending on data, it would only be logical to analyse the project data. Such analysis was done in parallel with the preliminary literature study as a pre-study. A healthy exercise that not only gave valuable insight and decision basis of what paradigm to chose, but also determined if it was possible to realise the project in the short time given for a master thesis.

The pre-study showed that the data set would not break any new size records. However, consisting of roughly 400 000 items ordered by ca 99 000 unique customers from ca 23500 unique products with ca 3800 unique styles, the data is still large enough to overload systems. Unfortunately, some data fields was not consistent over time. One such field was the technical description of items that lacked uniqueness and was rather sparse in appearance. Such inadequacy resulted in the decision of not pursuing the content-based paradigm as it is deemed to yield lesser result without proper data as its foundation.

Additionally, the argument for a knowledge-based paradigm is weak as it demands time consuming interaction. A paradigm perhaps better suited applications with a modest data set, usually defined by a low frequency of purchases (Jannach et al., 2011, p. 81). However, this is not the case for this project where the data possesses a high purchase intensity.

Fortunately, as the data analysis had ruled out two paradigms, the preliminary literature study gave positive indications for the collaborative filtering paradigm. The study revealed promising results toward context-aware recommender systems which mainly builds upon the collaborative filtering paradigm. Seemingly, it is also the most dominant paradigm after its performance in the Netflix prize challenge (Thompson, 2008) where a solution using factorization models and nearest neighbour won. Furthermore, the data analysis showed that the data set is large enough to support such paradigm.

With a reasonably strong decision basis, provided by the pre-study, collaborative filtering was selected as the ”best fit” paradigm to the project. Another insight from the preliminary literature study was the potential increase in prediction accuracy obtained by incorporating contextual awareness into recommender systems. It is an approach to capture the concept drift(see section 2.2) of customers. Due to the potential performance gains, it was decided that such a solution should be evaluated as well.

A custom-tailored implementation of the paradigm was deemed necessary as the specific project data departed from the established data set used in research. The data has a freshness factor, provided by the high throughput of items over a relatively short time period. Furthermore, a recommender system for Lindex will be forced to manage shopping carts which might contain several personalities, adding complexity to the problem. Additionally, customers will be affected by crowd psychology which suggest concept drifting is of importance and might increase the problem complexity further as contextual awareness will be introduced.

The task of implementing the project algorithm could not have been delegated to others as a straightforward blueprint could not be made without performance evaluations of different configurations, implementations and techniques. Therefore it was deemed to be a somewhat stumbling road to a prototype, needing a great deal of analysing skills. A task suitable for master thesis students of computer science.

In order to create the prototype, different techniques and theories had to be combined. The main theories were relevance feedback, SVD, k-means, a cart-personality-distribution algorithm and an item-ranking algorithm. The justification of using these techniques and theories is the following:

To construct the prototype, the most basic step, relevance feedback, was acquired; unfortunately, the lack of explicit feedback enforce the creation of implicit feedback. Such task is aggravated by the project constraint of only consider ongoing carts, thereby excluding useful information. Perhaps the obvious "easy fix" would be to revoke such constraint and consider all information sources. However, the constraints was set to ensure that the project scope would not be exceeded. It was established due to the time insufficiency a single semester brings. That notion, that complexity exhaust time resources, is a general presumption that also appears in this project. It have coloured many design choices, one of those choices is to use binary implicit feedback as it yields promising results and yet remains simple in nature. The relation it captures is the items a customer have bought; what it does not capture is the quantity of the item.

Unfortunately, the feedback is rather sparse and does not generate enough diffusion. Without the necessary amount of diffusion, the recommendations could become uniform. PCA is used to solve such issue. Essentially it amplifies diffusion by doing a basis change such that the PCA constraints is satisfied. Still, sparsity remains a problem. It is solved by reducing the dimension of the feature space such that redundant dimensions is discarded. That is, to only use a delimited set of the highest ranked principal components in an descending order. Such solution is derived from the intuition behind PCA: only the most relevant information will be used while the distortion is removed.

Still, PCA is primarily a concept. There are other techniques utilising the same concept; who yields other advantages. Such techniques is SVD and NMF which are preferred to PCA (Ricci et al., 2011, p. 45). SVD is a more general solution, allowing any matrix to be used (Shlens, 2005, p. 7). Unfortunately, it consumes considerable amount of resources in its computation which serves as an argument to pursue other techniques, such as NMF. However, evaluate the superior technique would not be possible in the short time frame given. Additionally, SVD always returns a unique solution which simplifies result evaluation. Thus, SVD was selected over the others.

Regardless of the techniques, a feature space was constructed. Then K-mean was applied to the space to obtain categories. An algorithm used widely for that purpose. It features simplicity and speed with a time complexity of exponential size (Vattani, 2011);

nonetheless, polynomial in practice (Arthur et al., 2009). k-means can be used with different similarities to obtain the clusters, all with their own advantages (see section 2.4). Unfortunately, cosine similarity could not be used in the prototype constructed in Matlab as some items was to closely located. Meanwhile, euclidean distance returned acceptable results and was therefore used over cosine. Of course there are other similarities to be used; unfortunately, time constraints did not allow to thoroughly test them and the literature did not indicate any major performance increase.

The produced categories from k-means is intended to reflect the personalities of customers. Moreover, a cart might consists of multiple personalities such as a skaters and a hip-hopper. Such diverseness, occurring seemingly sporadic, expresses a compound personality which usually describes multiple individuals such as a mother and her child etc. In order to better capture such compound personalities, the distribution of such categories is calculated. It would then be used to determine what categories the items should belong to, given a specific ongoing cart.

To increase prediction accuracy, a ranking of the items will be made relative to its cluster centroid. Again euclidean distance was used, mainly as it is the similarity selected in the k-means algorithm.

Theories, technique and how they are combined have now been explained and arguments for justification made. However, most of the decision is influenced by the results from evaluation which will be presented in later chapters. Thus, it is important to continue reading evaluation method etc.

3

Methods & Materials

This chapter presents the means necessary to realise the project, but also the structure of conduct and evaluation procedures. The chapter structure is as follows: First is the working process explained. Second, what materials where used. Third, the participants that made this project possible. Last, test procedures describing how the tests were conducted.

3.1 Procedure

The working procedure does mainly consists of an agile approach where weekly meeting with the supervisors is held to discuss progress and problems. There are three distinguishing steps/iterations:

1. The basics of the recommender system will be constructed such as SVD, similarities, K-mean, cart distribution and recommendation selection.

2. Create a end-product to be used in AB-testing at Lindex.

3. Develop context-awareness of the recommendation system then test and compare towards the basic system.

Step one and two were completed, but the last step could not be completed due to time limits. However, some results was made from step three and it was enough to conduct a discussion about it. The context-awareness step were actually supposed to occur before the end-product step, but to ensure a stable prototype for AB-testing were these two steps swapped. This could not wait as the final sprint between 3Bits and Lindex occured before summer.

3.2 Materials

No special hardware have been used other than simple workstations and a server, with the capacity of eight cores and 40 gb of ram. Their operating systems was Windows 8.1 and Windows Server R2. Other software that has been used is:

- Matlab
- Microsoft Visual Studio Ultimate x64 2013
- Doxygen
- Sharelatex
- Graphlab
- ConcurrentVisualizer

Matlab is a suitable program for prototype construction and evaluation as the broad supply of tools and support makes the development faster. Still, there are some drawbacks such as an excess of run-time errors. A result of not using a type-checker to catch some of those errors in compile time (Ranta & Forsberg, 2012). Furthermore, the software employs a non-free licence. Nonetheless, there exists an alternative, Octave, that is free and similar, but lacks some tools and support.

The licence demand made it necessary to create an implementation in a more suitable language after a working prototype had been forged. Such language was C# with .net. Featuring easy integration with depending systems. Furthermore, such experience does already resides inside 3bits, reducing the effort needed from the company to develop and maintain the code. Additionally, the Accord[1] framework has proved to be a useful extension that reduced development time, despite inexperience of the framework. It includes machine learning algorithms such as k-mean but also SVD etc, and is licensed under LGPL[2].

The C# implementations of the algorithms are slow in comparison to identical algorithms in Matlab. Matlab uses a variant called online K-means. It separates the execution into two phases called batch update and online update. The batch update phase uses ordinary Lloyd's batch K-means, describe in the theory chapter.

The second phase for ordinary online K-means selects a randomly chosen point and reassigns it to the closest neighbouring cluster. It then recalculates the centroids before the next iteration. However, the Matlab implementation does this step for every point individually, but only reassigns the displacement to the configuration that decreases the total sum of the distances the most.

These two phases are executed until convergence. In (Slonim, Aharoni, & Crammer, 2013) there is a corollary arguing that both online and batch returns at least local minimums. The Accord implementation uses Lloyds's K-mean as Matlab does but does

[1]http://accord-framework.net/

[2]https://www.gnu.org/licenses/lgpl.html

not have a phase two. However, Matlab's implementation is fairly stable while Accord:s differs in the second test phase.

The stability issue might not only be explained by a second phase but also the use of another initialisation. Matlab utilises a ”replicate” function which generates a given number of random initialisations and selects the configuration with the lowest total within-cluster point-to-centroid distances. The Matlab implementation in this project uses a ”replicate” of ten.

3.3 Participants in study

A sub goal of the thesis was to obtain experience from the industry by working for a company. This company is 3Bits[3]. They provides the workstations, help with integration and a supervisor, Jonas Hörnstein. He coordinated the work with 3Bits, gave valuable general guidance but also insights to how work in the industry is conducted.

The data set is provided by Lindex[4] as the system is a prototype for recommendations in their web-shop. In respect to their business and customers but also to follow the Swedish law of personal record[5], the information regarding the data set in any way is anonymous.

Obviously, Chalmers also played a part in this project. Providing a supervisor and examiner, Devdatt Dubhashi. Who supplied support such as literature suggestions, organizing reviews from institution personnel, personally reviewing report drafts and he also provided valuable guidance.

3.4 ItemRand- & frequent-algorithm

To obtain a reference point to the project algorithm was results from a lesser and simpler algorithm needed. These two algorithms was randomItem- and frequency-algorithm. The first, randItem-algorithm, selects a random item out of all the items and recommends it. The following pseudo code explains how it works: The second, frequency-algorithm,

Algorithm 7: randItem-algorithm

```
  Data: $items$, $itemsToRec$
1 for $i=1, i < itemsToRec, i++$ do
2     The i:th recommendations ← Randomly generated item from $items$
3 return recommendations
```

finds the most frequently ordered item and then recommends them in a descending order. The following pseudo code explains how it works:

[3]http://www.3bits.se/sv

[4]http://www.lindex.com/se/

[5]http://www.riksdagen.se/sv/Dokument-Lagar/Lagar/Svenskforfattningssamling/Personuppgiftslag-1998204_sfs-1998-204/?bet=1998:204

Algorithm 8: frequncy-algorithm

```
   Data: items, itemsToRec
1  for i=1, i < itemsToRec, i++ do
2      The i:th recommendations ← The i:th most frequent item from items
3  return recommendations
```

3.5 Test procedures

The test procedures is an important part of this project. An sequential approach to testing where conducted. More precisely, a waterfall where the first step is a basic test that examines if the chosen techniques is wise to use or if it is back to the drawing board. The second step simulates ”real-world” events using history data. If this step is passed than the third step will take place. The next step tests how context can be used to improve accuracy. The last test step was a AB-test phase where the procedure is described in the last section.

Basic test

The first test-phase of the project algorithm is a ten fold cross validation where the fold is constructed out of customers. The recommendation procedure is tested by extracting every cart of the test set and only consider those carts that contains at least two items. This test is constructed such that for every cart, one item A is randomly selected and separated from the rest of the cart. As an example, item A is then given to the recommender and in turn a new item X is recommended given item A.

Algorithm 9: Basic testing algorithm

```
Data: $carts = \{cart1 = \{itemA, itemB, itemC\}, cart2 \ldots\}$ , $customers$
1  $indices \leftarrow$ generate indices for a 10-fold crossvalidation with the size of $customers$
2  for  $k$=1; $k <= 10$; $k$++ do
3      $testCustomers \leftarrow customers( indices == k )$
4      $testCarts \leftarrow carts$ belonging to $testCustomers$
5      for  $i$=1, $i <$ size(testCarts), $i$++ do
6          $cart \leftarrow$ testCarts(i)
7          if  size(cart) $>= 2$ then
8              $selectedItem \leftarrow$ select a random item from $cart$
9              $restOfCart \leftarrow$ all items in cart that is not $selectedItem$
10             $randItem \leftarrow$ randItem-algorithm($selectedItem$)
11             $freqItem \leftarrow$ frequency-algorithm($selectedItem$)
12             $recItem \leftarrow$ project-algorithm($selectedItem$)
13             $errorRand = 0$ if item in $restOfCart$ is equal to $randItem$ else 1
14             $freqRand = 0$ if item in $restOfCart$ is equal to $freqItem$ else 1
15             $recRand = 0$ if item in $restOfCart$ is equal to $recItem$ else 1
16 return normalized errors
```

If any of the remaining items in the rest of the cart is equal to item X, the test is to be considered successful or otherwise a failure. That is, either one hundred percentage success or failure. The summary of all test for one fold will in worst case scenario return a number of failed carts equal to the number of carts given as input. The pseudo code for the test can be observed in algorithm 9, results and comparisons is presented in the result chapter 4.

Time test

The second test-phase is to train and test on two different sets of data, separated by time constraints. An example of such separation would be to train on all working days but test on the weekend. This test-phase better models real world execution where the algorithm will not be able to train on the future and test on the past as ten-fold cross-validation allows.

Further more, sets of sequential periods (table 3.1) will be referred to as categories in an attempt to increase the simplicity of forthcoming reasoning. These categories is defined by three columns: A period number, separated train and test set and the time intervals.

Category A		
1	Training Period	2013-03-26 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
2	Training Period	2013-03-27 00:00 — 2013-04-01 10:00
	Testing Period	2013-04-01 10:00 — 2013-04-01 12:00
3	Training Period	2013-03-28 00:00 — 2013-04-02 10:00
	Testing Period	2013-04-02 10:00 — 2013-04-02 12:00
4	Training Period	2013-03-29 00:00 — 2013-04-03 10:00
	Testing Period	2013-04-03 10:00 — 2013-04-03 12:00
5	Training Period	2013-03-30 00:00 — 2013-04-04 10:00
	Testing Period	2013-04-04 10:00 — 2013-04-04 12:00
6	Training Period	2013-03-31 00:00 — 2013-04-05 10:00
	Testing Period	2013-04-05 10:00 — 2013-04-05 12:00
7	Training Period	2013-04-01 00:00 — 2013-04-06 10:00
	Testing Period	2013-04-06 10:00 — 2013-04-06 12:00

Table 3.1: Testtime example

As the reader just have witnessed, the categories have an ungraceful size and is therefore located in the appendix.

The test will include three different algorithms to generate recommendation: The project algorithm, the randomItem-algorithm and the frequency-algorithm. For each test-iteration, every algorithm will have their recommendation tested towards the remaining customer cart.

Algorithm 10: Time testing algorithm given a *Single* item

```
Data: $carts = \{cart1 = \{itemA, itemB, itemC\}, cart2 \ldots\}$
for iter=0, iter < iterations, iter++ do
    for i=0, i < size of carts, i++ do
        $cart \leftarrow$ i:th cart of carts
        if size of cart >= 2 then
            for j=0, j < size of cart, j++ do
                $selectedItem \leftarrow$ j:th item of cart
                $restOfCart \leftarrow$ all items in cart that is not selectedItem
                $randItem \leftarrow$ randItem-algorithm
                $freqItem \leftarrow$ frequency-algorithm
                $recItem \leftarrow$ project-algorithm given selectedItem
                $errorRand = 0$ if item in restOfCart is equal to randItem else 1
                $freqRand = 0$ if item in restOfCart is equal to freqItem else 1
                $recRand = 0$ if item in restOfCart is equal to recItem else 1
```

The algorithm 10 explains in pseudo-code how the tests is conducted. Final results are the ration between the number of errors and the number of carts tested. However, item duplicates might occur in a shopping cart. Such situation occurs as a multiple purchase of the same item is done, socks etc. Therefore, it is not invalid to recommend the same item given to the recommender in these tests, but in a practical application should such behaviour be avoided as showing new items to the user is the objective of a real world recommender system. The tests on multiple items is described in algorithm 11.

Algorithm 11: Time testing algorithm given *Multiple* items

```
Data: $carts = \{cart1 = \{itemA, itemB, itemC\}, cart2 \ldots\}$, $nrItems$
for iter=0, iter < iterations, iter++ do
    for i=0, i < size of carts, i++ do
        $cart \leftarrow$ i:th $cart$ of $carts$
        for j=0, j < nrItems, j++ do
            int $oneItem = 0$;
            while oneItem == 0 AND givenItems contains oneItem do
                $rIndex \leftarrow$ create random index of $oneCart$ length
                $oneItem \leftarrow oneCart[rIndex]$
            $givenItems[j] \leftarrow oneItem$
        if size of cart >= nrItems then
            for j=0, j < size of cart, j++ do
                $selectedItem \leftarrow$ j:th $item$ of $cart$
                $restOfCart \leftarrow$ all items in cart that is not $givenItems$
                $randItem \leftarrow$ randItem-algorithm
                $freqItem \leftarrow$ frequency-algorithm
                $recItem \leftarrow$ project-algorithm given $givenItems$
                $errorRand = 0$ if item in $restOfCart$ is equal to $randItem$ else 1
                $freqRand = 0$ if item in $restOfCart$ is equal to $freqItem$ else 1
                $recRand = 0$ if item in $restOfCart$ is equal to $recItem$ else 1
```

Most test have been constructed such that the test set is no larger than two hours of data. This decision is an attempt to better test the real world situation that will occur in the AB-testing. The goal is to re-train the algorithm every two hours, thus the test set does not have to be any larger than two hours.

Context-aware tests

These test aims to evaluate how contextual information, incorporated in to the system, affects recommendation accuracy. Due to the lack of time was only pre-filtering tests conducted. Their structure is very similar to the time test (section 3.5) where the difference is the input set. There can be multiple input sets such as only Saturdays or Sundays. Another filtering constraint can be the location, the country as an example; however, such filters was not thoroughly tested and corresponding result can thus not be found in the result chapter 4.

Synthetic tests

The size for the Lindex datasets cannot be revealed which restrains the possibility to recreate and compare results. Therefore was a synthetic dataset created such that its size and distribution could be revealed.

The synthetic data size is 500 customers divided in to five group, containing 100 customer each. There are 50 unique items which is also divided into 5 groups with 10 items in each group. Every customer have one cart where each cart contains 5 items. The distribution of the cart is presented in table 3.2 where the number of items from each item group is presented for each customer group. The selection of specific items from a item group have been randomly generated for each cart.

Customer group	item groups {i1,i2,i3,i4,i5}
c1	4,1,0,0,0
c2	3,1,1,0,0
c3	1,0,0,3,1
c4	0,1,3,1,0
c5	0,0,2,3,0

Table 3.2: This is the cart distribution for the synthetic data.

The tests that were conducted on the synthetic data is similar to those done on the Time test. The procedure is thoroughly explained in algorithm 12.

Algorithm 12: Generate syntetic test set

```
Data: data, iterations, nrRepeats
1  for repeatProcess=0, repeatProcess < nrRepeats, repeatProcess++ do
2      allFolds ← create an array which stores the fold configuration
3      for i=0, i < size of data, i++ do
4          fold = Generate random folds of 10-fold-crossvaildation
5          allFolds[i] ← fold
6      for foldNr=0, foldNr < 10, foldNr++ do
7          trainingCarts ← Create a set of 9 folds
8          validationData ← Create a set out of the last remaining fold
9          for iter=0, iter < iterations, iter++ do
10             for j=0, j < size of cart, j++ do
11                 selectedItem ← randomly selected validationData carts items
12                 restOfCart ← all items in cart that is not selectedItem
13                 randItem ← randItem-algorithm
14                 freqItem ← frequency-algorithm
15                 recItem ← project-algorithm given trainingCarts
16                 errorRand = 0 if item in restOfCart is equal to randItem else 1
17                 freqRand = 0 if item in restOfCart is equal to freqItem else 1
18                 recRand = 0 if item in restOfCart is equal to recItem else 1
```

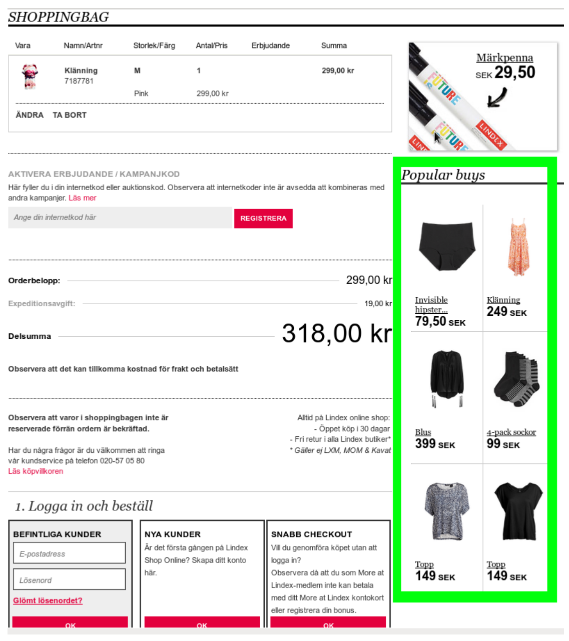
AB-testing

Most IT-systems is improved over time and to validate performance increase is AB-testing used. The AB-test works as follows: the new system is referred to as A and the old system is referred to as B. One of these systems, either A or B but not both, is presented to the users by random selection and the performance is recorded over time which will reveal if the new system have increased performance.

In Lindex web-shop there are two AB-tests. The first one is located at the check-out page, table 3.3 , and the second one is located at the product page, figure 3.1. The product-page test displays three products while the checkout page test presents six products.

The checkout page test has incorporated the project-algorithm as its A-system and no recommendation services as the B-system. Table 3.3 presents the A-system at the left side and the B-system at the right side. Figure 3.1 visualizes the product-page test where the project-algorithm is the A-system and the old Lindex recommendation services is the B-system.

The results from the AB-testing where created using the t-Stat test, where the standard errors are compared to see what hypothesis testing (A or B) has the most positive effect.



SHOPPINGBAG

Vara	Namn/Artnr	Storlek/Färg	Antal/Pris	Erbjudande	Summa
	Klänning 7187781	**M** Pink	**1** 299,00 kr		**299,00 kr**

ÄNDRA TA BORT

Märkpenna
SEK 29,50

AKTIVERA ERBJUDANDE / KAMPANJKOD
Här fyller du i din internetkod eller auktionskod. Observera att internetkoder inte är avsedda att kombineras med andra kampanjer. Läs mer

Ange din internetkod här REGISTRERA

Orderbelopp: 299,00 kr
Expeditionsavgift: 19,00 kr
Delsumma 318,00 kr

Observera att det kan tillkomma kostnad för frakt och betalsätt

Observera att varor i shoppingbagen inte är reserverade förrän ordern är bekräftad.

Har du några frågor är du välkommen att ringa vår kundservice på telefon 020-57 05 80
Läs köpvillkoren

Alltid på Lindex online shop:
- Öppet köp i 30 dagar
- Fri retur i alla Lindex butiker*
** Gäller ej LXM, MOM & Kavat*

1. Logga in och beställ

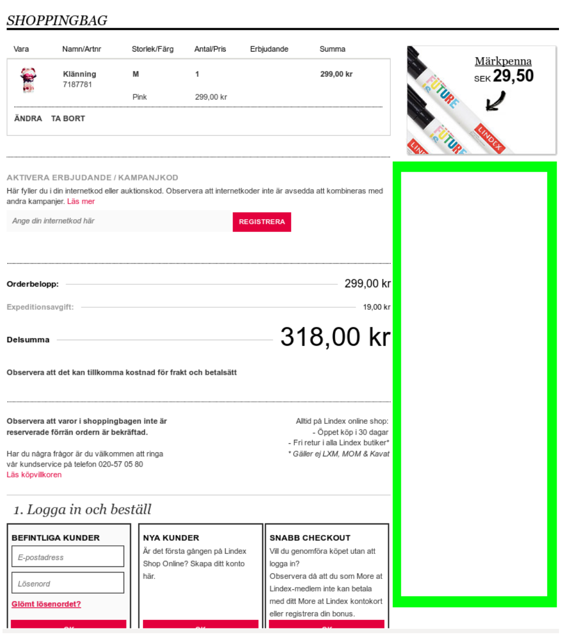
BEFINTLIGA KUNDER
E-postadress
Lösenord
Glömt lösenordet?
OK

NYA KUNDER
Är det första gången på Lindex Shop Online? Skapa ditt konto här.
OK

SNABB CHECKOUT
Vill du genomföra köpet utan att logga in?
Observera då att du som More at Lindex-medlem inte kan betala med ditt More at Lindex kontokort eller registrera din bonus.
OK

Popular buys

Invisible hipster... 79,50 SEK	Klänning 249 SEK
Blus 399 SEK	4-pack sockor 99 SEK
Topp 149 SEK	Topp 149 SEK



SHOPPINGBAG

Vara	Namn/Artnr	Storlek/Färg	Antal/Pris	Erbjudande	Summa
	Klänning 7187781	**M** Pink	**1** 299,00 kr		**299,00 kr**

ÄNDRA TA BORT

Märkpenna
SEK 29,50

AKTIVERA ERBJUDANDE / KAMPANJKOD
Här fyller du i din internetkod eller auktionskod. Observera att internetkoder inte är avsedda att kombineras med andra kampanjer. Läs mer

Ange din internetkod här REGISTRERA

Orderbelopp: 299,00 kr
Expeditionsavgift: 19,00 kr
Delsumma 318,00 kr

Observera att det kan tillkomma kostnad för frakt och betalsätt

Observera att varor i shoppingbagen inte är reserverade förrän ordern är bekräftad.

Har du några frågor är du välkommen att ringa vår kundservice på telefon 020-57 05 80
Läs köpvillkoren

Alltid på Lindex online shop:
- Öppet köp i 30 dagar
- Fri retur i alla Lindex butiker*
** Gäller ej LXM, MOM & Kavat*

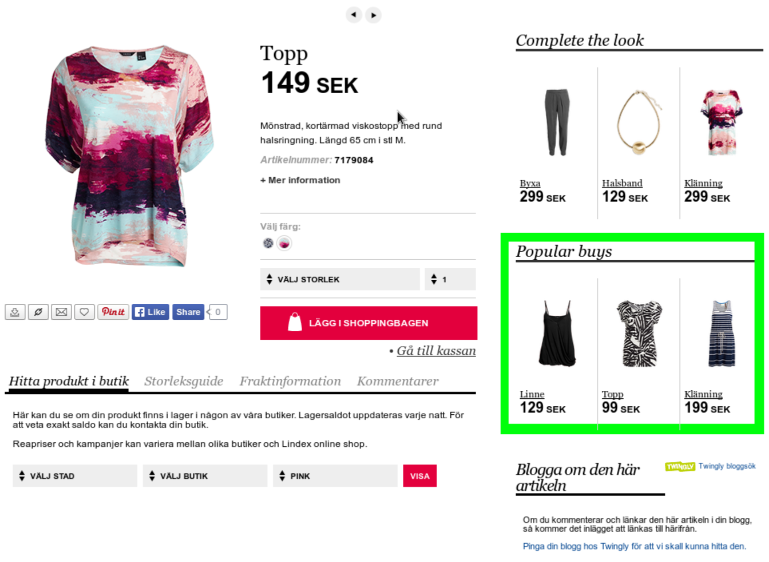
1. Logga in och beställ

BEFINTLIGA KUNDER
E-postadress
Lösenord
Glömt lösenordet?
OK

NYA KUNDER
Är det första gången på Lindex Shop Online? Skapa ditt konto här.
OK

SNABB CHECKOUT
Vill du genomföra köpet utan att logga in?
Observera då att du som More at Lindex-medlem inte kan betala med ditt More at Lindex kontokort eller registrera din bonus.
OK

Table 3.3: The recommender system location in the web shopping cart. Both cases are marked in green, to the left our recommendations and the right the empty case.



Topp
149 SEK

Mönstrad, kortärmad viskostopp med rund halsringning. Längd 65 cm i stl M.
Artikelnummer: **7179084**
+ Mer information

Välj färg:

VÄLJ STORLEK 1

LÄGG I SHOPPINGBAGEN

• *Gå till kassan*

Hitta produkt i butik *Storleksguide* *Fraktinformation* *Kommentarer*

Här kan du se om din produkt finns i lager i någon av våra butiker. Lagersaldot uppdateras varje natt. För att veta exakt saldo kan du kontakta din butik.

Reapriser och kampanjer kan variera mellan olika butiker och Lindex online shop.

VÄLJ STAD VÄLJ BUTIK PINK VISA

Complete the look

Byxa 299 SEK	Halsband 129 SEK	Klänning 299 SEK

Popular buys

Linne 129 SEK	Topp 99 SEK	Klänning 199 SEK

Blogga om den här artikeln

Twingly bloggsök

Om du kommenterar och länkar den här artikeln i din blogg, så kommer det inlägget att länkas till härifrån.

Pinga din blogg hos Twingly för att vi skall kunna hitta den.

Figure 3.1: The recommendations location on the product pages is marked in green. This is how it looks like for the project algorithm and Lindex old algorithm.

4

Result

THE structure of the result is divided into five parts: basic-, time-, context-, synthetic- and AB-tests. Basic tests reports the test result from the prototype created in Matlab. Time tests reports the test results of how the configuration of the C#-implementation was evaluated where most of the result is presented in figures and tables as so is for most sections. Context tests reports the result of different time contexts as for example training on morning hours. The synthetic test is conducted to compensate for not revealing data set sizes which hopefully will give the reader a better reference. The AB tests report the results of how the algorithm worked online in a real world setting.

4.1 Basic tests

The result presented here are from the MatLab-implementation and is generated for each period in a category or many categories. As an example $\{A,B,C\}$ is the concatenation of the categories A, B and C and is referred to as many categories. The result is then the average over all periods in that category of the concatenation of categories. A period in an category is referred to as $A : x$ where x is the period. All categories can be found in appendix A. An example on how the results look are depicted in the Table 4.1. The results are from the MatLab-implementation and are configured by five variables: the training data set size and test data set size that is referred to as $Setsize$, the number of recommendations, number of iterations, clusters size and number of dimensions. The data set sizes will actually not be revealed as there exists an nondisclosure agreement that says so, but also to preserve the integrity of Lindex and their customers. To compensate for the lack of data set sizes was the synthetic data set created, the result from a test with that data set is presented later in this chapter 4.4. However, a percental difference between the training data set sizes is presented in section 3.5, Time test, to enable a discussion of the sizes as they are very important for the recommendation accuracy.

The cluster size, *clusters*, presented in table 4.1 was chosen out of observations when constructing the experimental prototype(for an explanation of the variables, see section 2.1). The number of dimension SVD reduces the implicit feedback into is presented as *dim*. *Cart*, *Freq* and *Random* is the label for the project-, frequency and randItem-algorithm results. *rec* is simply the number of items that are going too be recommended. In the test was *rec* set to ten as that number was suggested from 3bits. The number of iterations, *Iter*, is how many times the tests where done to generate more stable results.

Category	Set size	#rec	Iter	Clusters	Dim	Cart	Random	Freq
$D:1$	$-\times-$	10	20	1100	10	31,34%	4,07%	25,26%
$D:2$	$-\times-$	10	20	1100	10	31,33%	4,08%	25,22%
$D:3$	$-\times-$	10	20	1100	10	30,88%	3,95%	24,69%
$D:4$	$-\times-$	10	20	1100	10	30,06%	4,41%	23,37%
$D:5$	$-\times-$	10	20	1100	10	30,17%	4,33%	25,29%
$D:6$	$-\times-$	10	20	1100	10	31,11%	4,37%	25,08%
$D:7$	$-\times-$	10	20	1100	10	31,48%	4,43%	24,90%
D	$-\times-$	10	20	1100	10	30,91%	4,23%	24,83%

Table 4.1: Results obtained from 10-fold crossvalidation category D, defined in appendix A

Figure 4.1 is a line chart of table 4.1 that visualizes the deviations within the category. An emphasize on the result of each period in category D, instead of an average.

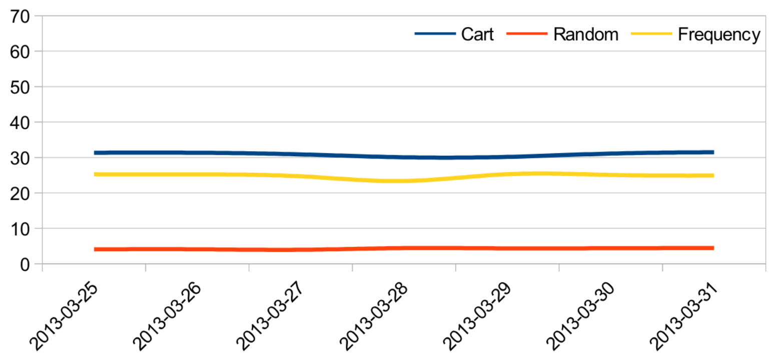


Figure 4.1: Results of 10-fold cross validation on the periods from category D

4.2 Time tests

The results of the time tests will mainly be presented as tables of diagrams with corresponding captions and explanations. The following results are generated from the time testing phase described in chapter 3.

Category	A	B	C	{ A,B,C }
Train size	—	—	—	—
Test size	—	—	—	—
#Items to rec	10	10	10	10
Iterations	500	500	500	500
Clusters	1100	1100	1100	1100
Dimensions	10	10	10	10
Result Cart	36.46%	38.72%	37.19%	37.46%
Result Random	2.82%	2.85%	3.34%	3.00%
Result Freq	23.36%	25.75%	28.09%	25.74%

Table 4.2: Results obtained from execution of different categories, defined in appendix A

There are three different results in table 4.2: First, the result from the project algorithm, referred to as Cart. Second, the result generated from the randomItem-algorithm, referred to as Random. Last, the result generated by the frequency-algorithm, referred to as Freq. The result represents the average success rate for each algorithm over its different time periods, as is explained in the test procedure section.

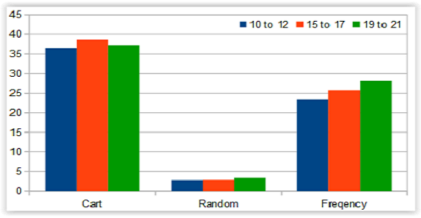


Figure 4.2: The performance of the three algorithms on average for a testing set over two hour at three time periods.

As in the previous chapter the configuration parameters had to be evaluated for the C#-implementation. To do such evaluation a number of tests where performed such that the following parameters could be determined: cluster size, dimension size, training size, testing size, number of items to recommend, number of test iterations. Figure 4.2 is a bar diagram of table 4.2 using the sets $\{A,B,C\}$. It illuminates the vague difference between the sets in the project-algorithm and how the reverse relation emerge in the frequency-algorithm. Variation in the result of the project-algorithm differs which is an important note in the algorithmic comparison, how much is well described in the later parts of this chapter. It occur as the test will not yield the exact same results each iteration due to the random nature of k-mean. Still, the project-algorithm obtains over 10% better results in contrast to the frequency-algorithm.

As mentioned before the test and train size are neglected. However, it is still possible to discuss the percent differences between periods in categories etc. A matrix that compares categories can be observed in table 4.3.

	Sep	Oct	Nov	Dec	Jan	Feb	Mars	April
September	100%	10%	-10%	-22%	36%	45%	-12%	-6%
October	-9%	100%	-18%	-29%	24%	33%	-20%	-14%
November	11%	22%	100%	-13%	51%	62%	-2%	4%
December	28%	40%	15%	100%	74%	86%	13%	20%
Januari	-27%	-20%	-34%	-43%	100%	7%	-35%	-31%
Februari	-31%	-25%	-38%	-46%	-6%	100%	-39%	-35%
Mars	14%	24%	2%	-11%	55%	65%	100%	7%
April	6%	17%	-4%	-17%	45%	55%	-6%	100%

Table 4.3: Comparison of the data size between months too give a intuition of how the data size effect the results. In the table it can be seen that each month size is divided by the other months sizes so when looking at each row it represents how much larger a month is compared to the others.

Importance of cluster size

In order to optimize results, the cluster size had to be dynamic. Still, the question what size to select remained. To obtain the optimal cluster size tests were made and the result is presented in Figure 4.3. The different sizes tested was 50% up to 90% of unique item size with a 10% step size on the categories *A*,*B* & *C*.

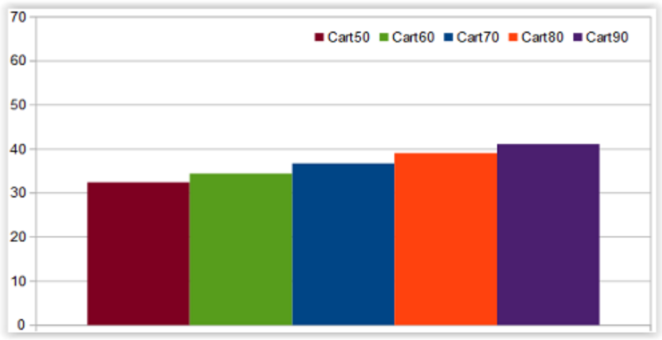


Figure 4.3: Visualizes the relation between prediction accuracy and the number of clusters used. The meaning of Cart50 is that the project algorithm is used with a cluster size of 50% of unique item size. Cart60 have the same meaning except with 60% instead and so on.

Figure 4.3 reveals a near linear relation between cluster size and accuracy. A dynamic cluster size of 80% was chosen.

Importance of dimensions

The dimension size needed when doing reduction in SVD had to be determined. Thus, the following figure was generated:

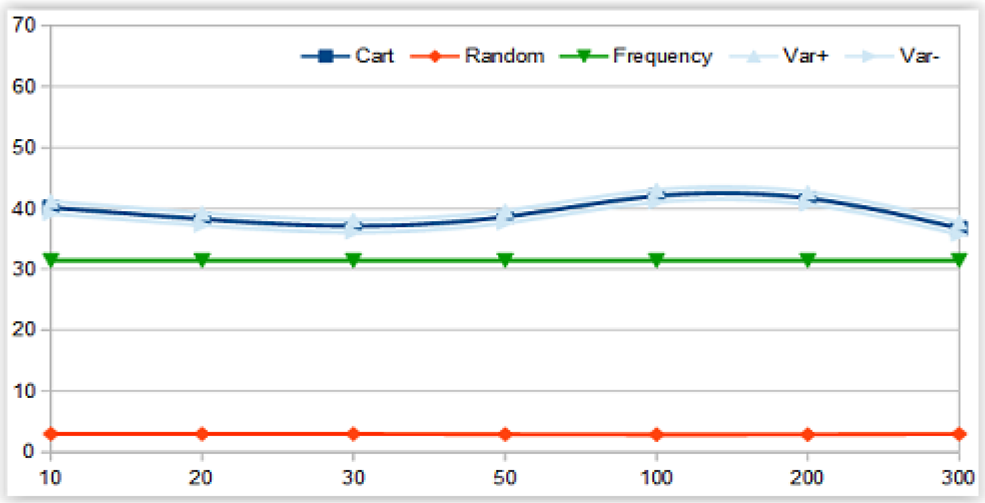


Figure 4.4: The performance change as the dimensions increase. Performance at the y-axis and dimension size in the x-axis.

Figure 4.4 does not establish any significant result which in itself reveals that increasing the dimension size will not increase performance significantly, the test used the P:1 time category. Consequently, a dimension size of 10 was selected.

Importance of training time

As the dimension and cluster size was established, the natural step was to determine the training size needed to obtain satisfactory results. The following figures represents the performance for the different training sets which reveals the "best fit" size for the project-algorithm. The first figure presents the result given a single item while the second presents the result given multiple items.
Table 4.4 presents results on how the system is affected by using different training sizes. The variance, given a single item, is 0.68% and 0.25% for the project- and randItem-algorithm. The variance, given multiple items, is 1.54%, 0.7% and 0.46% for the project-, frequency- and randItem-algorithm. The result is increase with an increase training set size; however, the increase is not significant enough to justify the increased amount of resources needed to re-train as it gives an increased time span between re-trainings which will decrease accuracy over time. Thus the shortest training set, one week, was chosen.

Categories Y, given a *single item* Categories Y, given a *multiple item*

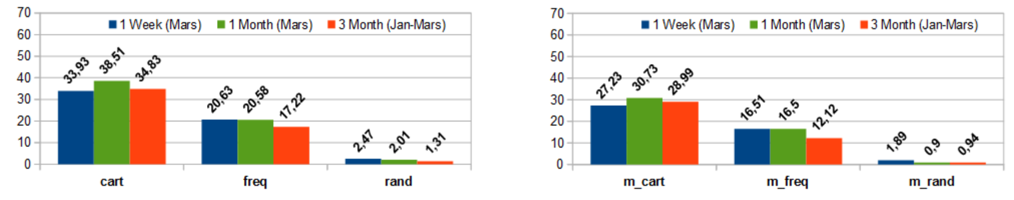


Table 4.4: Results using different training sizes

Additionally, a test where made that decreased the training set size. The table 4.5 reflects how results is affected by decreasing the number of sequentially following training days, resulting in an enormous size difference of the training sets: 1980,74%. Still, such difference is to be expected as the ration between the dates are 14:1.

Single: Categories N,O and P *Multiple*: Categories N,O and P

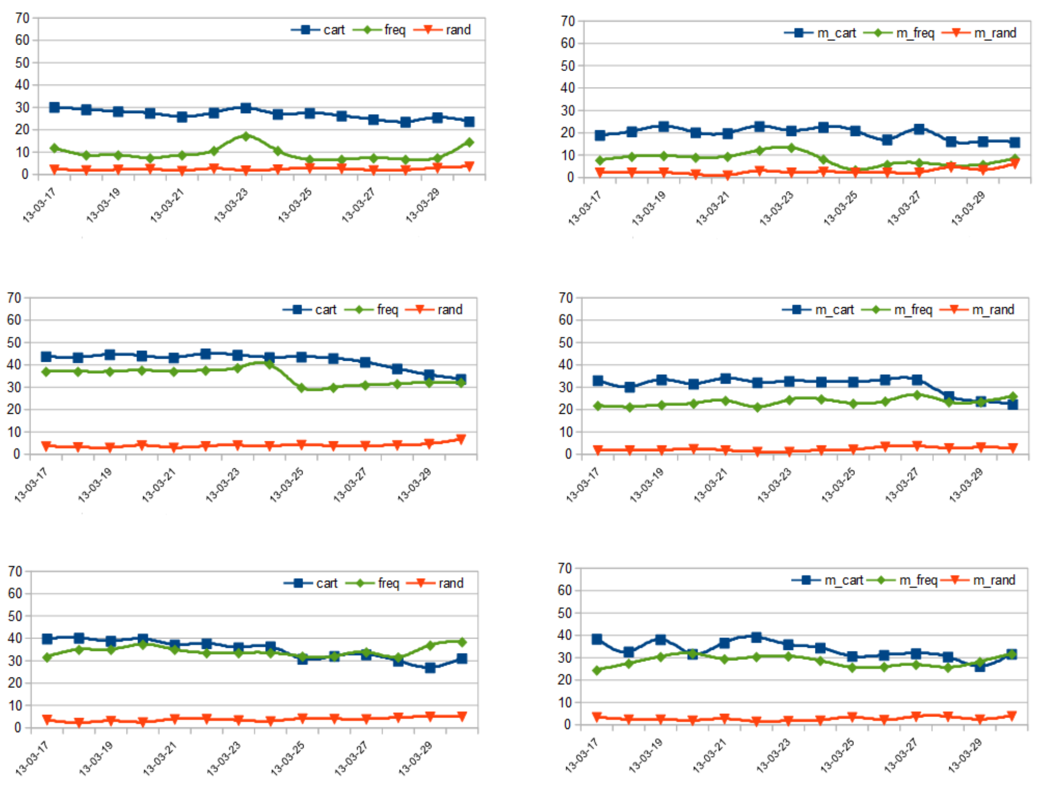


Table 4.5: The table shows decreasing training times from 14 days to 1 concerning categories N, O and P.

Time shift testing

In order to verify stability of the project algorithm over time, test with similar training and testing set was performed and is presented here, but with a slight shift of time.

The first test was to run the "shift" tests, training and testing with the same time parameter except that they are shifted one day forward. The test result from shift test are represented as figures in a 2×3 table. Each row will present results from a specific category, usually defined above the figures. The right side result is generated from "time testing algorithm given a single item" and the left side "time testing algorithm given multiple items". However, these two uses the different test algorithms (see section 3.5).

The y-axis represents the percentage of successful recommendations and the x-axis is the period start of the test, but with hours excluded(see appendix A for the exact times). Furthermore, the lines represents the success gradient between different periods and the markings is the exact result given a period. There are different markings for the algorithms such that results can be interpreted in black and white as well. In turn the tables structures remains the same but differentiates in months such as A, B and C represents different periods in March:

12-Oct	12-Nov	12-Dec	13-Mar
E, F & G	H, I & J	K, L & M	A, B & C

Table 4.6

As previously mentioned, the results have a variance. In the figures of the right half from table 4.7 is the variance: {1.18%, 1.02%, 1.07%} for the project-algorithm, given the categories {A, B, C}, and {0.83%, 0.88%, 0.73%} for the randItem-algorithm. There is no variance for the frequency-algorithm in the test given a single item as they are always the same, no randomization involved. In contrast, the variance for tests given multiple items is {2.48%, 2.43%, 2.32%} for the frequency-algorithm. Furthermore, {4.32%, 3.43%, 3.59%} and {1.68%,1.72%, 1.69%} for the project-algorithm and randItem-algorithm.

Single: Categories A,B and C *Multiple*: Categories A,B and C

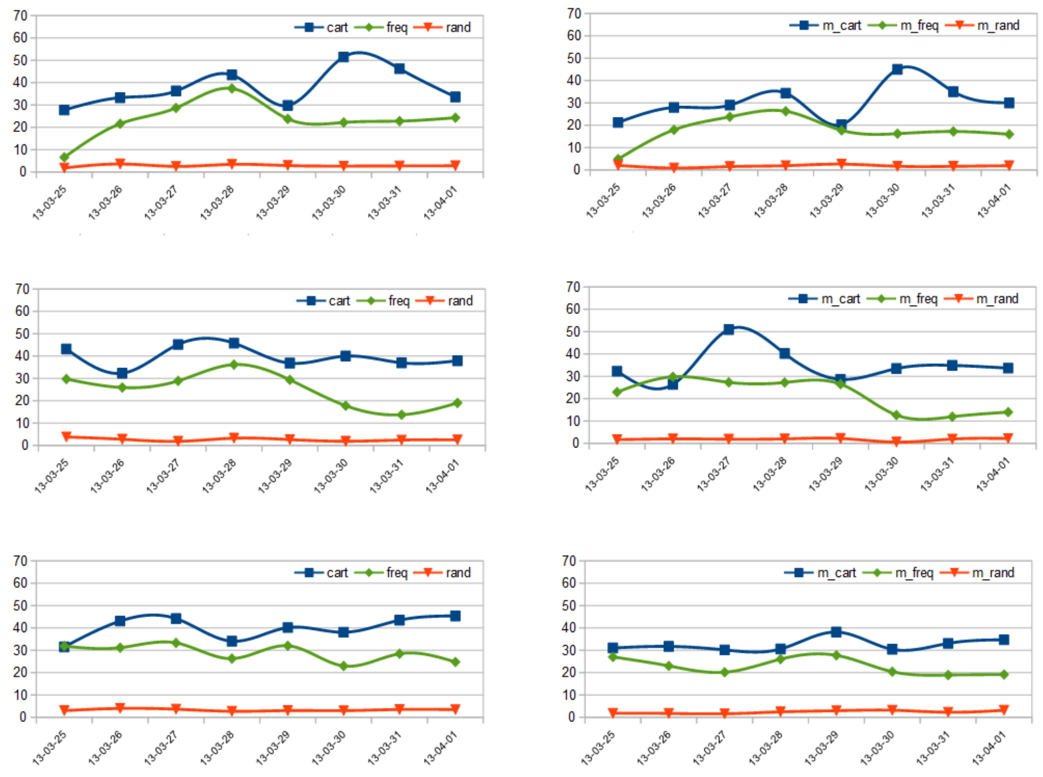


Table 4.7: The left side of the table visualizes the results of recommendations made on shifting data sets of categories A, B & C. The right side is the results of the recommendations made given two items, still the same categories. Observe that the dates of the diagrams are the first day of the training set.

Ten iterations was used to obtain these results. More iterations is preferable; unfortunately, the test requires a lot of computational resources which is not justifiable as the accuracy increase of the test is slim in comparison. Furthermore, ten recommendations was made in each test as a similar configuration with multiple recommendations would most likely be used in a real world application. Additionally, some items will be filtered out in a real world application as some items has already been presented to the user etc.

Table 4.8 have the same configuration as table 4.7 except for the time categories. The variances for categories $\{E, F, G\}$ given a single item is: $\{1.49\%, 1.97\%, 1.33\%\}$ and $\{1.32\%, 1.35\%, 1.23\%\}$ for the project- and randItem-algorithm. Given many items: $\{5.80\%, 5.59\%, 5.44\%\}$, $\{3.39\%, 3.56\%, 3.02\%\}$ and $\{2.73\%, 2.63\%, 3.03\%\}$ for the project-, frequency- and randItem-algorithm.

Single: Categories E,F and G *Multiple*: Categories E,F and G

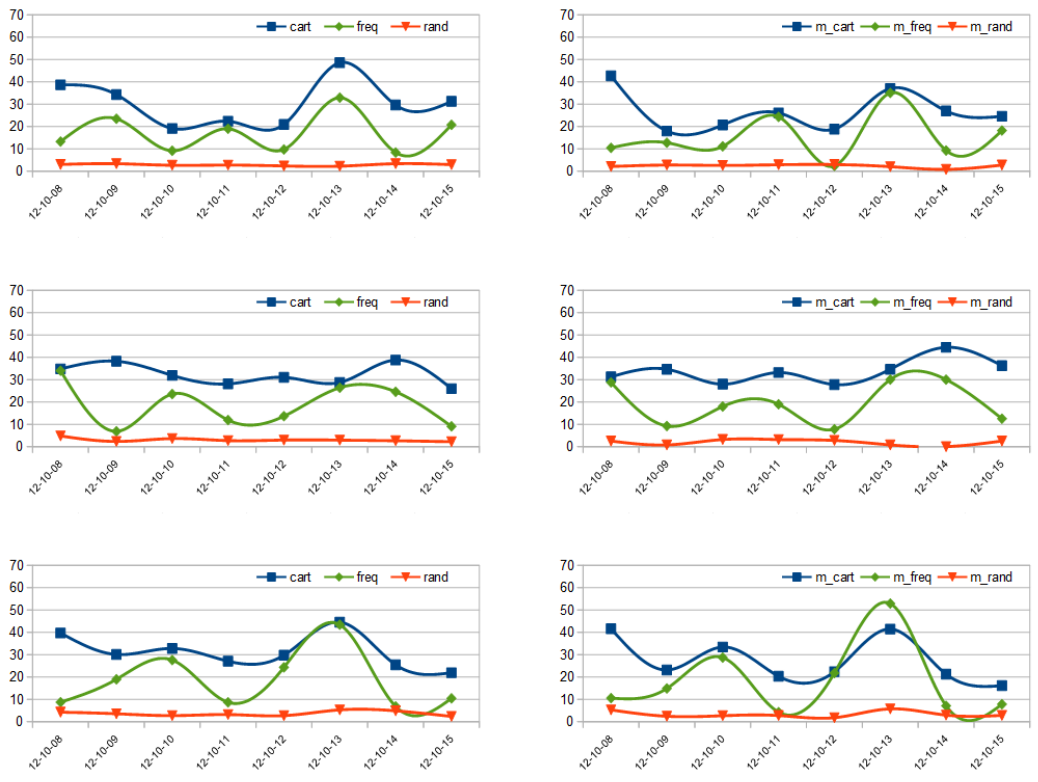


Table 4.8: The left side of the table visualizes the results of recommendations made on shifting data sets of categories E, F & G. The right side is the results of the recommendations made given two items, still the same categories . Observe that the dates of diagrams are the test day.

Table 4.9 have the same configuration as table 4.7 except for the time categories. The variances for categories $\{H,\ I,\ J\}$ given a single item is: $\{1.81\%,\ 1.75\%,\ 1.35\%\}$ and $\{1.35\%,\ 1.34\%,\ 1.07\%\}$ for the project- and randItem-algorithm. Given many items: $\{6.5\%,\ 6.26\%,\ 4.88\%\}$, $\{4.16\%,\ 2.89\%,\ 2.85\%\}$ and $\{3.24\%,\ 3.25\%,\ 2.36\%\}$ for the project-, frequency- and randItem-algorithm.

Single: Categories H,I and J *Multiple*: Categories H,I and J

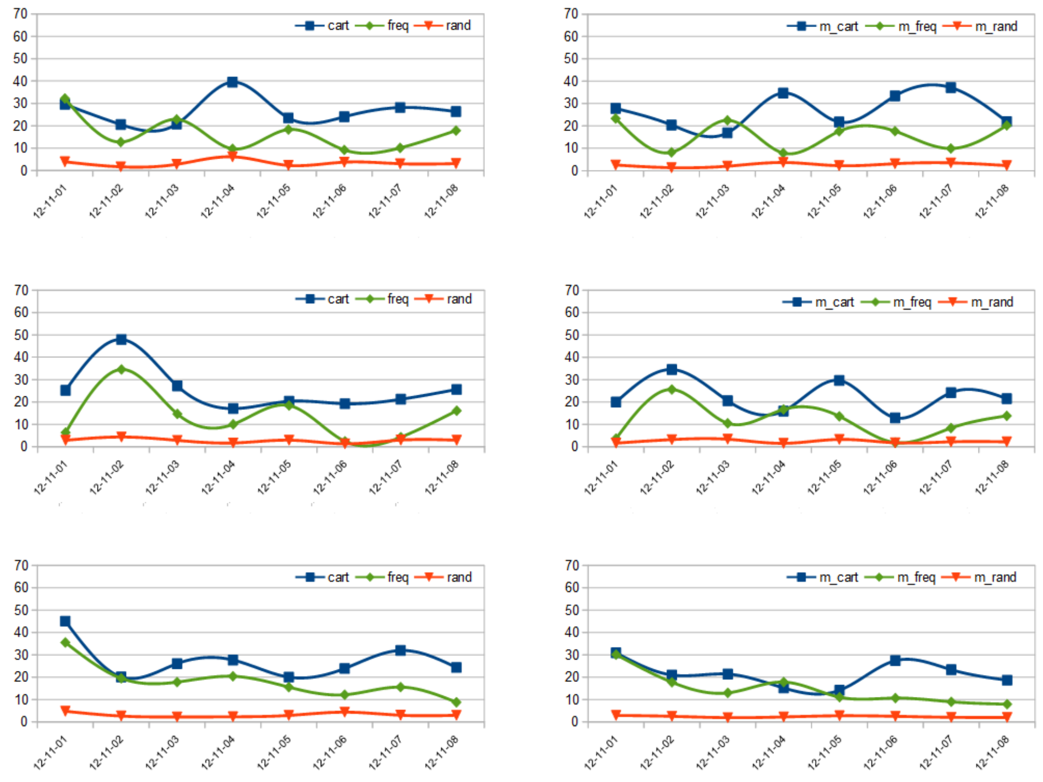


Table 4.9: The left side of the table visualizes the results of recommendations made on shifting data sets of categories H, I & J. The right side is the results of the recommendations made given two items, still the same categories. Observe that the dates of the diagrams are the first day of the training set.

Table 4.10 have the same configuration as table 4.9 except for the time categories. The variances for categories $\{K,\ L,\ M\}$ given a single item is: {1.20%, 1.51%, 1.03%} and {0.87%, 1.02%, 0.76%} for the project- and randItem-algorithm. Given many items: {4.84%, 4.70%, 3.67%}, {2.87%, 2.31%, 2.23%} and {1.75%, 1.16%, 1.72%} for the project-, frequency- and randItem-algorithm.

Single: Categories K,L and M *Multiple*: Categories K,L and M

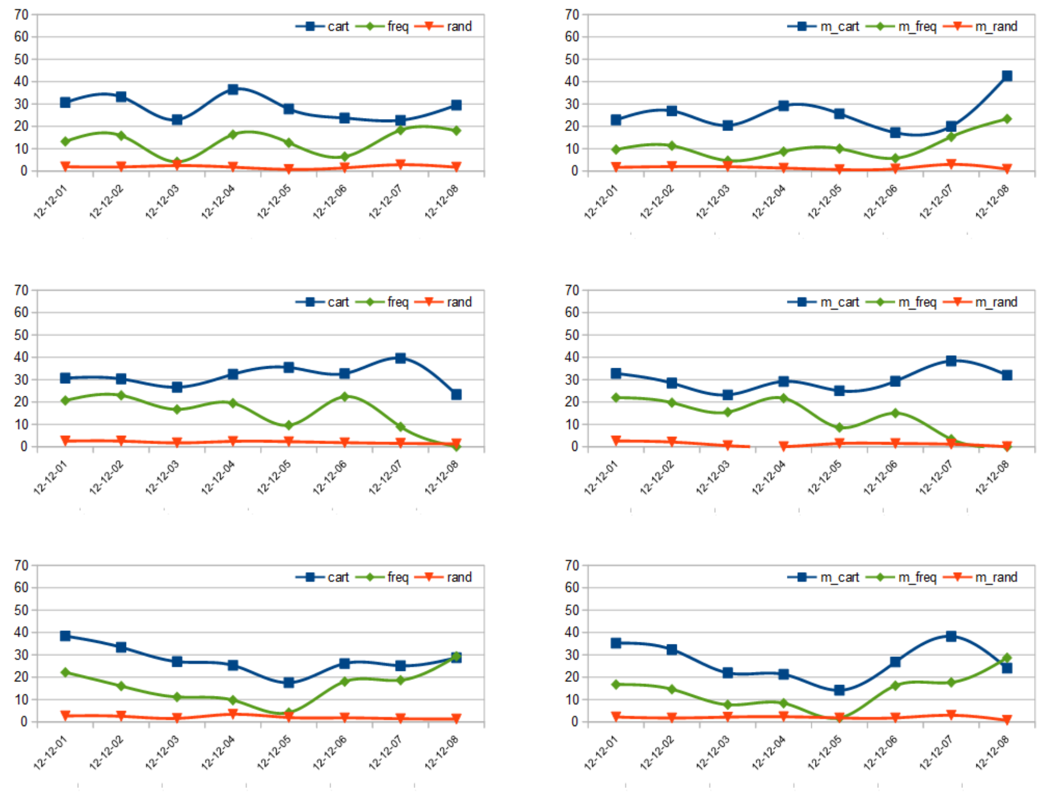


Table 4.10: The left side of the table visualizes the results of recommendations made on shifting data sets of categories K, L & M. The right side is the results of the recommendations made given two items, still the same categories. Observe that the dates of the diagrams are the first day of the training set.

The difference between November and December is interesting as the sales is expected to increase in December, which it does, and the recommender might respond differently to the increased intensity of sales. The difference is presented in table 4.11 & 4.12.

Category	H	K		I	L		J	M	
Carts	25.94%	25.55%	-0.39	23.41%	27.77%	4.36	24.52%	24.33%	-0.19
Random	3.34%	2.18%	-1.16	2.74%	2.11%	-0.63	3.23%	2.27%	-0.96
Frequency	16.61%	13.09%	-3.52	13.31%	15.06%	1.75	18.13%	16.12%	-2.01

Table 4.11: Compares the average result from categories visualized in table 4.9 and 4.10 for a *Single* item.

Category	H	K		I	L		J	M	
Carts	26.78%	25.57%	-1.21	22.38%	29.77%	7.39	21.48%	24.00%	2.52
Random	2.59%	1.52%	-1.07	2.37%	1.13%	-1.24	2.33%	0.67%	-1.66
Frequency	15.93%	11.07%	-4.86	11.74%	13.21%	1.47	14.64%	28.67%	14.03

Table 4.12: Compares the average result from categories visualized in table 4.9 and 4.10 for *Multiple* items.

Table 4.13 have the same configuration as table 4.9 except for the time categories. The variances for categories {Q, R, S} given a single item is: {1.61%, 1.74%, 1.21%} and {1.19%, 1.25%, 0.88%} for the project- and randItem-algorithm. Given many items: {8.14%, 5.17%, 3.84%}, {2.57%, 3.76%, 2.65%} and {1.91%, 2.95%, 2.08%} for the project-, frequency- and randItem-algorithm.

Single: Categories Q,R and S *Multiple*: Categories Q,R and S

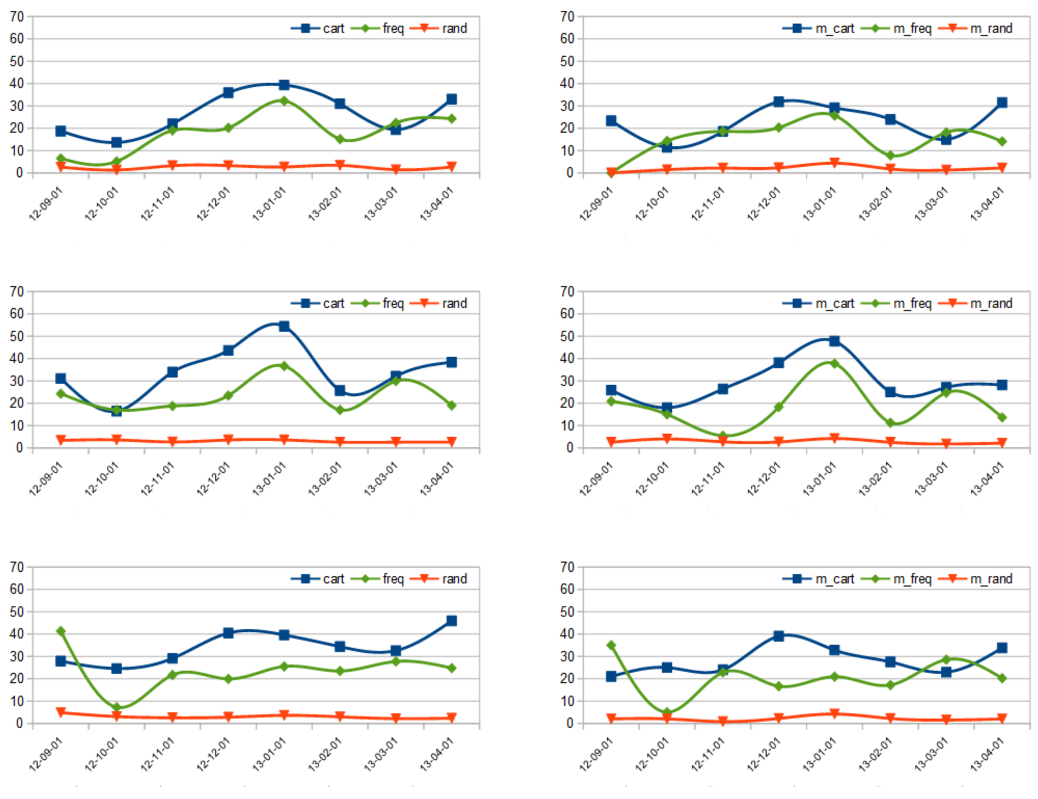


Table 4.13: Presents the result form training on the first seven days and test on the last for the categories Q, R and S.

The maximum size difference of the training sets in categories Q,R & S is 212,09%

which indicate that some months have an higher intensity than others. This is also depicted in the size difference of table 4.3. As can be seen, some sets sizes differs a lot such as December with a size 86% bigger than in January.

Category	A	B	C	E	F	G	H	I	J
Carts	37.69%	39.74%	39.95%	30.58%	32.19%	31.40%	26.54%	25.43%	27.36%
Frequency	23.36%	25.03%	28.81%	17.06%	18.75%	18.60%	16.61%	13.31%	18.13%
	14.33	14.69	11.14	13.52	13.44	12.80	9.93	12.12	9.23

Table 4.14: Compares the average result between Carts and Frequency from Category A-J , given a *Single* item.

Category	K	L	M	N	O	P	Q	R	S
Carts	28.34%	31.36%	27.66%	26,7%	41,76%	34,81%	26.17%	34.46%	33.42%
Frequency	13.09%	15.06%	16.12%	9,34%	34,69%	34,06%	17.71%	24.77%	23.70%
	15.25	16.30	11.54	17.36	7.07	0.75	8.46	9.69	9.72

Table 4.15: Compares the average result between Carts and Frequency from Category K-M , given a *Single* item.

Table 4.14 and 4.15 compares the results between the project- and frequency-algorithm. The green text under the percentage is their difference. If it is red it means the frequency have obtained a better overall result for that particular category. The same comparison is done in tables 4.16 & 4.17 for multiple items.

Category	A	B	C	E	F	G	H	I	J
Carts	30.33%	35.03%	32.45%	26.83%	33.76%	27.44%	26.78%	22.38%	21.48%
Frequency	17.46%	21.51%	22.78%	15.41%	19.40%	18.47%	15.93%	11.74%	14.64%
	12.87	13.52	9.67	11.42	14.36	8.97	10.85	10.91	6.84

Table 4.16: Compares the average result between Carts and Frequency from Category A-J , given *Multiple* items.

Category	K	L	M	N	O	P	Q	R	S
Carts	25.57%	29.77%	26.73%	19,6%	30,6%	33,34%	22.67%	28.81%	27.57%
Frequency	11.07%	13.21%	13.94%	8%	23,25%	28,21%	14.48%	18.63%	20.16%
	14.50	16.56	10.17	11.6	7.35	5.13	8.19	10.18	7.41

Table 4.17: Compares the average result between Carts and Frequency from Category K-M , given *Multiple* items.

4.3 Context tests

This section households the results of contextual pre-filtering testing. The tests is equal to those of Time-test(see section 3.5) but takes multiple and different input sets that is explained in their categories.

Category	Carts	Frequency		Random
T:1	22.45%	18.75%	3.70	4.27%
T:2	11.28%	8.02%	3.26	3.52%
T:3	12.18%	10.51%	1.67	3.77%
A:7	46.19%	22.73%	23.46	2.67%
B:7	36.93%	13.68%	23.25	2.36%
C:7	43.42%	28.40%	13.23	3.46%

Table 4.18: The first three periods presents results for contextual pre-filtering while the bottom three presents one weeks execution with the same test set

Category	T:1	A:7		T:2	B:7		T:3	C:7	
Carts	22.45%	46.19%	-23.74	11.28%	36.93%	-25.65	12.18%	43.42%	-31.24
Random	4.27%	2.67%	1.60	3.52%	2.36%	1.16	3.77%	3.46%	0.31
Frequency	18.75%	22.73%	-3.98	8.02%	13.68%	-5.66	10.51%	28.40%	-17.89

Table 4.19: Compares the same algorithm with and without contextual pre-filtering. When the non-contextual algorithm out-perform its contextual counter part, a red negative unit represents their difference. The opposite is represented as a green positive unit.

Table 4.18 and 4.19 have the following configuration: A cluster size of 80%, 10 dimensions, 100 iterations and a variance, given a single item, of {0.28} and {0.39} for the project- and randItm-algorithm. The variance given multiple items is {1.14}, {0.83} and {0.75} for the project-, frequency- and randItem-algorithm.

Category	Carts	Frequency		Random
U:1	16.09%	16.57%	-0.48	3.22%
U:2	14.28%	13.85%	0.43	2.94%
U:3	28.28%	26.86%	1.42	3.38%
A:8	33.55%	24.26%	9.29	2.72%
B:8	37.85%	18.97%	18.88	2.46%
C:8	45.34%	24.73%	20.61	3.39%

Table 4.20: The first three periods presents results for contextual pre-filtering while the bottom three presents one weeks execution with the same test set

Category	U:1	A:8		U:2	B:8		U:3	C:8	
Carts	16.09%	33.55%	-17.46	14.28%	37.85%	-23.57	28.28%	45.35%	-17.07
Random	3.22%	2.72%	0.50	2.94%	2.46%	0.48	3.38%	3.39%	-0.01
Frequency	16.57%	24.26%	-7.69	13.85%	18.97%	-5.12	26.86%	24.73%	2.13

Table 4.21: Compares the same algorithm with and without contextual pre-filtering. When the non-contextual algorithm out-perform its contextual counter part, a red negative unit represents their difference. The opposite is represented as a green positive unit.

Table 4.20 and 4.21 have the following configuration: A cluster size of 80%, 10 dimensions, 100 iterations and a variance, given a single item, of {0.81} and {0.96} for the project- and randItm-algorithm. The variance given multiple items is {3.24}, {2.70} and {2.12} for the project-, frequency- and randItem-algorithm.

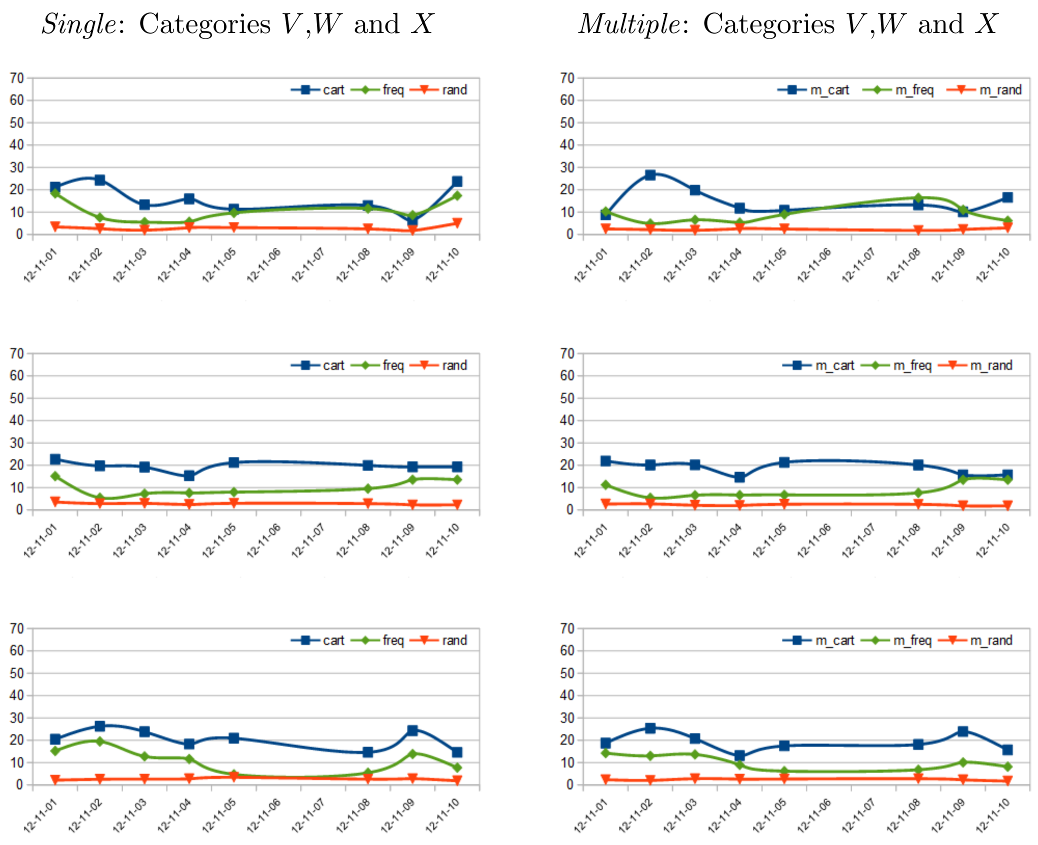


Table 4.22: The figures visualizes the results from test on either the morning, lunch or evening of the same day(See appendix for exact description). The training set consists of 15 sequential weekdays of the same time period of the day, mornings etc. Consequentially, 12-11-06 and 12-11-07 does not yield any results as they are not weekdays.

The configuration of Table 4.22 is one hundred iterations, 80% cluster size, 10 dimensions, output of ten recommendation and given one respectively two items. Furthermore, the variance for singular recommendations is {1.35%,0.97%} and {1.05%,0.78%} for the project- and randItem-algorithm. Multiple recommendation have a variance of {4.13%,3.50%}, {2.93%,1.73%} and {2.51%,1.74%} for project-, frequency- and randItem-algorithm.

4.4 Synthetic testing

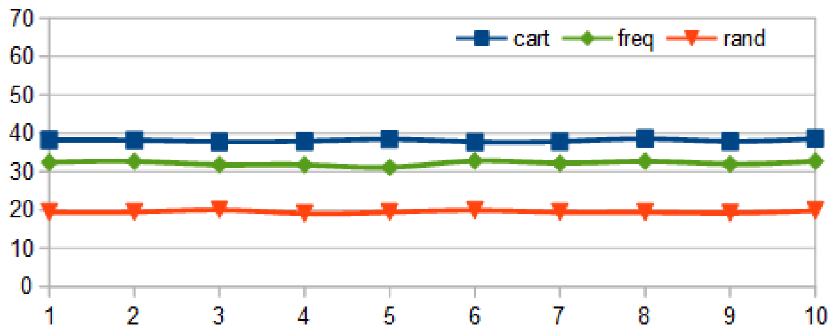


Figure 4.5: Results of 10-fold cross validation on synthetically constructed data, given a *single item.*

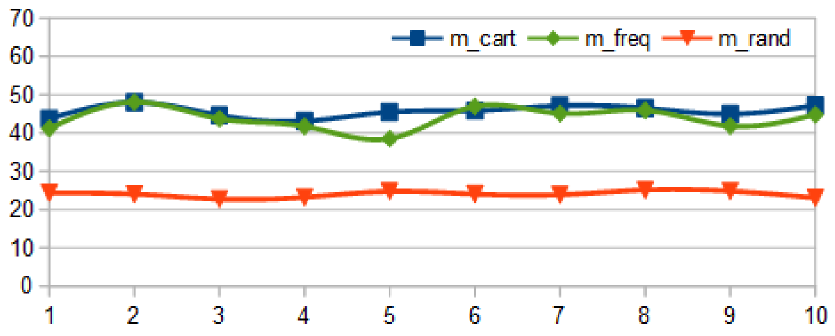


Figure 4.6: Results of 10-fold cross validation on synthetically constructed data, given *multiple items.*

The configuration for figure 4.5 and 4.6 is one hundred iterations with an additional ten iteration for each fold, 80% cluster size, 10 dimensions, output of ten recommendation and given one respectively two items. Furthermore, the variance for singular recommendations is 5.33%, 4.3% and 3.56% for the project-, frequency- and randItem-algorithm. Multiple recommendation have a variance of 21.90%, 22.22% and 18.67% for project-, frequency- and randItem-algorithm.

4.5 AB-tests

The results gathered from the AB-tests are divided in two cases: product page and checkout page.

4.5.1 Product Page

The old Lindex recommendation system is the A-test and the B-test is the project-algorithm. In these tests have different values been measured where the most important ones are the average value of an order and the number of recommended products in an order. The average order value for the A-test was slightly better than the B-test, 0,87% better, but is not statistical valid as can be seen from the t-test values in table 4.23.

Each order contained 0,83 recommended products in the A-test, while the B-test only contained 0,59 recommended products per order. The t-test results, located in table 4.24, shows that the values are statistical valid. The tests where conducted between week 32 to 36 in the year of 2014.

t-stat	0.810523507
P(T<=t) one-tail	0.208824516
t Critical one-tail	1.644930037
P(T<=t) two-tail	0.417649032
t Critical two-tail	1.960082944

Table 4.23: t-test for ordervalue: Two sample assuming unequal variance

t-stat	13.70917
P(T<=t) one-tail	$7.09 * 10^{-43}$
t Critical one-tail	1.644933
P(T<=t) two-tail	$1.42 * 10^{-42}$
t Critical two-tail	1.960087

Table 4.24: t-test for recommended products per order: Two sample assuming unequal variance

4.5.2 Checkout Page

The A-test did not contain any recommendation while the B-test used the project-algorithm to present recommendations. In these tests did the average order value increase with 0.75% using recommendations from the project-algorithm. Unfortunately are these values not statistical valid as the difference is to small compare to the number of samples taken, see table 4.25.

t-stat	-0.69386
P(T<=t) one-tail	0.243889
t Critical one-tail	1.644932
P(T<=t) two-tail	0.487778
t Critical two-tail	1.960085

Table 4.25: t-test for ordervalue: Two sample assuming unequal variance

5

Discussion

This chapter will first do a general discussion of the results presented in the chapter 4. The general discussion will follow the structure of Result and mainly consists of interpretations and insights made. After the general discussion will the discussion of the research question take place. The insights obtained from the general discussion is necessary to give an answer to the research questions.

Discussion of Basic Tests

The results from cross-validation in Matlab mainly serves as an indicator to how future results of the project algorithm might turn out. Indeed, the next test phase tends to render similar margins between the project algorithm and the most frequent algorithm. Actually, table 4.2 appears to have a similar margin as in table 4.1, but elevated five percentage. Another interesting observation is that the result from randItem-algorithm is decreased with about one percentage, probably due to the reduction of the test set. Such behaviour could be explained by the fact that randomly selecting an item out of all items in a bigger set, and then hope to find the same item in an much smaller set, is not very likely.

Discussion of the Test Configuration

The configuration for the project algorithm such as cluster size, dimension size and etc. had to be evaluated and determined. Results of such evaluation are located in figure 4.4 and 4.3 where no significant differences between the dimension sizes can be observed. It might seem strange as the dimension reduction is large, around 100-1000 times. The obvious choice is to use a dimension as small as possible to shorten execution times. Moreover, the cluster size have a fairly linear increase. Still, 80% cluster size is preferred as more clusters takes time to compute and the increase is not significant. From these result it was decided a dimension size of ten and cluster size of 80% of unique items size should be used as most categories is not larger than two weeks.

Discussion of the Time Test, given a ***Single Item***

To better understand and predict how the algorithm would behave in a real world situation, a test featuring the same circumstances had to be made. This test has been named time test (see chapter 3.5). Interestingly enough, some periods used in this test seems to generate better results than the sequentially next period. Such behaviour might seem a bit strange; however, the same phenomenon occurs in the most frequent algorithm as well. In some diagrams the two algorithms even have and inter-operating oscillation. To some extent such behaviour is explained by how the cold-start problem is managed by the project algorithm. However, by providing a small test set, much of the items is bound to already be apart of the training set. Hence, the situations where a cold-start situation occurs is limited.

It is also important to realise that categories A, B and C is merely a shift of some hours. Still, their results can be very different which reveals how important the test set is. The same notion might also explain the difference between periods in the same categories. An exaggerated example is when a test set only contains the most rare occurring items in the training data. Such items will rarely be recommended and is definitely not among the most frequent occurring items. It might not be the most prestigious explanation but hard test sets yields poor results.

Additionally, when analysing periods with a higher separation of time for the project algorithm such as in table 4.5, it becomes obvious that some periods yields better result consequentially over all categories. The ascendancy between November and December is particularly prominent. Still, there is a strong descendance from January to March and again an ascendancy between March and April. However, an increase in sales before Christmas is to be expected and might explain the increase of prediction accuracy in December. Interestingly enough, the frequency algorithm does not differentiate as much compared to the results between January and March. Such behaviour suggest the sale are more diverse in their nature for December but still includes pattern detectable by the project algorithm.

A plausible explanation for the increased patterns detectability might be the size difference in the training set. It is increased by roughly 64% from November to December. The same behaviour is observed between March and April where the frequency algorithm decreases and the project algorithm increases its accuracy. Yet again, training size difference is increased, this time by roughly 24%.

Such behaviour suggest that much data in a small time span increases the project algorithms accuracy compared to the frequency algorithm. It might seem far fetched but it could be compared to the Nyquist sampling theorem where to few sample does not recreate the true behaviour of a signal[1]. In this situation, the signal would be the user patterns.

Additionally, the project algorithm performs poorly in category P from table 4.5.

[1]`http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem`

As it turns out, the test size is roughly 36% larger for that category compared to the others. Furthermore, the training size is decreasing between the different periods and all three categories indicate a decrease in accuracy when the training size is reduced below six days. Such indicators serves as arguments for the existence of a relation between test size and training size. However, it does not establish any arguments that increased intensity of the test set would reduce the project algorithms accuracy.

If the three different categories is compared to each other it becomes obvious that there is a reduction in the accuracy of the project algorithm relative to the frequency algorithm. Depending on the perspective taken, it could instead be interpreted as an increase of accuracy for the frequency algorithm. Additionally, the accuracy for the project algorithm is increased from category N compared to category P, but the frequency algorithm increases more. Such behaviour suggests the test data is not diverse and contains many most frequent items. Perhaps that tells some about the users behaviour as category P only includes purchases done in the evening? However, such behaviour cannot be observed over a years span in table 4.5 which, on the contrary, suggests that category P might contain an unfortunate test set.

Such notion begs the question: does categories with a larger time difference yield significant diverse results? The averages in table 4.11 indicate no such thing. The greatest divergence of the project algorithm is a little larger than four percentage which is not much as a variance of roughly 1.5% exists. Furthermore, the frequency algorithm have a divergence that exceeds one and a half percentage in table 4.11 for all three comparison which suggest the different data set are somewhat similar over time. However, the exact difference can be studied in the visuals of table 4.9 and table 4.10.

To recap what has been said:

- The frequency- and project-algorithm inter-operates to some extent, suggesting the project-algorithms result differs accordingly to the behaviour of the dataset.
- More intensive periods seems to reveal user patterns better and in turn yield better result for the project-algorithm.
- There exists a relationship between training size and test size.
- The randomItem-algorithm yields low results for relatively big training sets.

Discussion of algorithmic comparison, given a *Single Item*

Table 4.14 compares shifting categories and so does table 4.15 for categories K,L & M. These tests returns a unequivocally result: the project-algorithm outperform the frequency-algorithm. Not only at specific periods but over many periods, both with smaller intervals and larger. Additionally, the difference of results is roughly 11.2% better in average which indicates the project-algorithm finds and uses user patterns such that its accuracy is improved compared to the frequency-algorithm.

However, if one look more closely on the results, all evening categories return less difference than the previous. Such behaviour is also revealed in table 4.15 for the categories

Q, R & S where a decrease of difference occur from mid-morning toward the evening. Fortunately, so does not the recommendation accuracy for the project-algorithm. It is simply the frequency-algorithm that have increased in accuracy.

Discussion of the Time Test Phase, given *Multiple Items*

Up until now have only results from recommendation given a single item been discussed. However, results from recommendation given two items has also been tested. One such test is visualized in table 4.12. The results show an decrease in accuracy for both the frequency- and project-algorithm. However, that is to be expected as there is one item less to compare the recommended items against. That is, a 25% reduction of possible comparisons is made on average as the average cart size is five. Furthermore, the multiple-items-test utilizes random selection of items to the recommender which increases the variance of the results.

Interestingly enough, the first three periods of H,I & J yields similarly pattern to its single-item counterpart. However, the patterns is somewhat attenuated and the sequentially following periods have an increased divergence. This is not the case for the frequency-algorithm which maintains the pattern structure with some variations. It suggests the project-algorithm still captures user pattern and beats the frequency-algorithm. However, it is not possible to strictly compare the results between the two tests as they are distinctively different (see section 3.5).

Discussion of algorithmic comparison, given *Multiple Items*

The results from table 4.16 and 4.17 does also reveal positive results toward the project-algorithm; however, somewhat attenuated as has been stated in the previous section. Such performance indicates that the results is conclusive, the project-algorithm outperforms the frequency with the current data and setting.

Discussion of Behaviour and Context

Previous observations suggests that the users change their behaviour toward the evening as they purchase the more frequent occurring items. Perhaps, working people with children does not have the time to conduct e-commerce until the evening when the kids has gone to bed. Their particular behaviour is probably also influenced as their children will need cloths as well. Thereby rises another issue, the growth of children such that they will need larger cloths on a frequent basis. The huge expense associated with that issue cannot motivate greater diversity of cloths such as trends etc.

Additionally, people that work might not have the same energy needed in the effort of shopping. In that situation it might be easier to follow campaigns or sales event as they tend to give the perception of being a good choice. Furthermore, an important factor is the current clothing trends for grown ups. People that work can usually afford trend cloths while unemployed's might have to settle with cheaper alternatives. Of course there is a presumption that trend cloths is more expensive in this argument and might not agree with reality in all situations. However, the factor behind the effect is probably the same, namely crowd psychology[2]. Given such hypothesis it might be reasonable to only train on data from one country or geographical region that house hold the same trends etc.

Unfortunately, as time is such a rare commodity, it was not possible to do a thorough evaluation of such interesting suggestion given the time frame of a master thesis. Still, it reveals territories of contextual pre-filtering that is of interest to further studies given various hypothesis of psychology. Perhaps somewhat data specific to the business in clothing e-commerce as the data is very fresh and have a high throughput of items. Furthermore, one of the most important aspect of the data set is the emphasis on shallowness. It might not be the most benignant word but it do express the lack of knowledge of the item such as material qualities, workability and frequency of usage. It might sound strange but customers prioritise what they see mixed with what they think other will think of them with such items. It sometimes results in purchases of items uncomfortable to the user but they are still willing to pay a lot of money for such items.

A more practical example would be occasionally clothing such as a dress to celebrate new year etc. It is not the most workable clothing, it is expensive and the lack of usage suggest it might not be the best purchase. Still, almost everyone have such clothing and usually of expensive trend brands. Moreover, as the frequency of such item is low, it is hard to capture in a recommender system which further argues for consideration of contextual information.

Discussion of Contextual Pre-filtering

When testing contextual pre-filtering, categories U & T, it was quickly discovered that some tests did not return good results such as $U : 1$ in table 4.20. Additionally, the period is very sparse on data, the training set is only 4 times larger than the test set.

Another important observation is that the difference between the frequency- and project-algorithm is very small which suggest a bad performance for the project-algorithm. Furthermore, the randomItem-algorithm generates good results compared to non-contextual test as can be observed in table 4.19 and 4.21. Such behaviour has already been explain, there exists to little training data in comparison to the test size.

As the observant reader have already noticed does the project-algorithm outperform its contextual counterpart in table 4.19 and 4.21. Interestingly enough, so does the frequency-algorithm as well. Such behaviour is explained with the same answer: there exists to little training data in comparison to the test size.

Synthetic testing

Figure 4.5 is very similar to the cross-validation test done for the matlab prototype, presented in figure 4.1. The difference between the project- and frequency-algorithm is almost the same but their result is elevated with roughly 8%. This is somewhat to be

[2]`http://en.wikipedia.org/wiki/Crowd_psychology`

expected as the data is synthetically generated and does not contain any special user pattern or concept drifts. However, the test given two items, whose results can be located in figure 4.6, is very different and have a very large variance, 22%, for both the project- and frequency-algorithm in comparison to a single item configuration that have roughly 5% for the project-algorithm, 0% for the frequency-algorithm.

This suggest that the current distribution equation for many items is unstable. Still, it is important to remember that the test cases is not the same and the result is therefore not entirely comparable but rather a reference. However, the frequency-algorithm performs equally to the project-algorithm which suggest the project-algorithm yields unsatisfactory results given multiple items. This indicates that a lot of improvements can be made for the distribution computations given multiple items.

Another important observation is the rough 8% increase of results compared to the experimental Matlab prototype. This can be explained by the fact that the basic test selects items from a cart by random where the time test iterate through all items and thus have no randomization involved in the test-phase. Furthermore, the implementations of the different techniques is different such as k-mean which can be another factor for the higher variance and the elevated results.

Discussion of AB-testing

It is obvious that the project-algorithm is outperformed by the current recommendation service at Lindex by the number of item recommended. However, the order-value is almost the same which is somewhat surprising as the Lindex recommendation service recommends 41.03% more items per order but does not significantly improve the order-value compared to the project-algorithm.

The increase is in the number or orders containing recommended products which suggests the project-algorithm predict user patterns but probably only for customers with a clear purchasing pattern. These shortcomings is a direct consequence of not utilizing all available data, but instead enforce constraints on data collection.

Moral Perspective

After talking a lot about how peoples virtual data can be used to achieve different goals, it might be wise to stop and ask if there are some moral issues regarding this kind of engineering. The system handles personal data and constructs recommendations based on it. Thus, these recommendation is personal and directed to the user, if the user then leave the computer and someone else starts using it, s/he will then be in the position where observation of the recommendations can be made.

By observing the recommendations it could be possible to decipher the previous users preference about the items. It might not be very harmful in most cases but if it is taken to the extreme there might be complications. An example: a male have been shopping female cloths, such actions might reveal personal preferences that can go against the norm or reveal potential female relations that he would like to keep a secret. Perhaps a more practical example would be a situation were the user has purchased presents and than the person, for whom the presents where intended, uses the computer and sees the recommendation that clearly reveals the intentions.

Furthermore, cookies could be used to enhance the recommendations by gather a lot of information from different areas that might not only be web-shop specific, given the cookies is accepted. Additionally, if the database is compromised, all the data could potentially be view by anyone. Is it morally right to use the data in this way, given these issues? What can be done to increase security?

To answer the question, careful consideration of the business and the information gather have to be done. In this case there are not very many impacts toward the user if the recommendations is revealed to others. However, there are some potential high impact on relational basis but with very low probability. Thus, the benefit to the masses out-weight the few that can actually suffer from miss-usage.

Fortunately, it is possible to protect the user integrity by considering every new cart as a new user if they have not logged in. This prevents others from getting access to the personal recommendations, given the user have logged out etc.

Research question

The research questions will be repeated here in order to refresh the readers memory:

Given a user that puts an item into his/her shopping cart, is it possible to create an implicit-feedback based collaborative filtering recommender system from shopping carts history that is interchangeable with a explicit-feedback based CF RS? How does this affect the performance in Neighborhood and factor models? Can the system be enhanced with contextual information and system addictiveness in such way that it will improve performance, perhaps out perform a non-CARS?

The first question is a bit hard to give a straight yes/no answer as such test were not conducted. The goal was to compare the project-algorithm to Graphlab, but that goal was down prioritized in order to ensure a working prototype for AB-testing. Of cause such test would have been valuable in the evaluation before an AB-test but an AB-test will give true performance results and is therefore prioritized. A comparison to a recommender system using explicit feedback can be done afterwards if further interests exists. Still, with high uncertainty, the comparison would probably not differentiate much.

The answer to the second question is that the restrains, on what information is allowed to be used to construct an recommender system, is not advised if the best possible recommendation accuracy is sought. However, fairly good accuracy can be obtain with those restrains and it simplifies the system such that the resources needed to do the computations is decreased.

In order to answer the last question, one have to implement adaptiveness and thoroughly evaluate performance on contextual awareness models and not just pre- and post-filtering. As this was not done, cannot a straight answer be given. However, contextual pre-filtering did not increase prediction accuracy given the project data and constraints.

6

Conclusion

THE thesis aim was to construct and evaluate a collaborative filtering recommender systems given data from Lindex. Such system was constructed and evaluation concerning performance on historical data was done. Even some test on contextual pre-filtering where performed, but most importantly were AB-testing done on Lindex web-shop.

Fortunately, the first evaluations returned unequivocal conclusive results: the project-algorithm outperform the frequency-algorithm. A joy-giving outcome of the Time tests. However, the complexity of the solution might have been unnecessary and simpler solution could have sufficed. To obtain such understanding, a comparative evaluation has to be done. Unfortunately, time did not allow such test to be performed.

Another area that needs more attention is the data set. A comparative evaluation toward other sets would be in order as the solution could be biased toward the specific data set. Unfortunately, it is hard to find good and *free* data sets that contains a cart-based structure. A "good" data set have a high intensity of orders in a relatively short time period. If such set was available, it would have been possible to reveal the information, sizes etc. That information is important for future research.

The most surprising results was perhaps the low success rate of contextual pre-filtering where an increase in recommendation accuracy was to be expected from the pre-study. Unfortunately, the results says otherwise. However, the evaluation is not as thorough as it could be and an increase is probably still possible.

Future work

As previously said, a more complex model to capture the concept drift should be thoroughly evaluated. There are some interesting context-awareness models such as higher order SVD (Karatzoglou, Amatriain, Baltrunas, & Oliver, 2010, p.3) that could be a candidate. Such model could go well in-hand with an more adaptive approach such as incremental SVD(Brand, 2002). As the adaptive approach increase the periods between re-trainings of the feature space. However, higher order SVD will be more computational intensive which indicates there might not be any gains of computational resources. The most likely benefits would be increased recommendation accuracy.

Other areas of interest would be to incorporate more information in to the system such as:

- Use URL:s visited by the user and other user to generate output data
- Use search terms created by the user and other users to generate output data
- Measure time spent on particular items
- Interpret trend blogs using natural language or RDFa[1] to gain knowledge of what items might become popular in the future

Furthermore, different psychology hypothesis, mentioned in the discussion chapter, could be useful in the quest of increasing recommendation accuracy. Particularly in the paradigm of collaborative filtering as it tries to capture what categories of a person the user are. Then improvement could be made if it is possible to foresee how customers would react using models based on theories from psychological science.

[1] `http://en.wikipedia.org/wiki/RDFa`

Bibliography

Adomavicius, G., & Jannach, D. (2013, March). Preface to the special issue on context-aware recommender systems. *The Journal of Personalization Research*, *152*($10.1007/s11257 - 013 - 9139 - 2$), 125-136.

Adomavicius, G., & Tuzhilin, A. (2005, June). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(1), 734-749. doi: 10.1109/TKDE.2005.99

Arthur, D., Arthur, D., Manthey, B., Manthey, B., Roglin, H., & Roglin, H. (2009). k-means has polynomial smoothed complexity. In (p. 405-414). IEEE. Retrieved from `www.summon.com`

Austin, D. (2009, August). *We recommend a singular value decomposition.* American Mathematical Society. (`http://www.ams.org/samplings/feature-column/fcarc-svd`)

Bazire, M., & Brézillon, P. (2005). Understanding context before using it. *CONTEXT 2005, LNAI 3554*, 29-40.

Brand, M. (2002). Incremental singular value decomposition of uncertain data with missing values. In (Vol. 2350, p. 707-720). Berlin, Heidelberg: Springer Berlin Heidelberg.

G Linden, J. Y., B Smith. (2003, January). Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, *7*. doi: 10.1109/MIC.2003.1167344

Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2011). *Recommender systems : an introduction.* Cambridge University Press.

J. S. Milton, J. C. A. (1986). *Introduction to probability and statistics.* Mc Graw Hill.

Kalman, D. (1996, January). A singularly valuable decomposition: The svd of a matrix. *The College Mathematics Journal*, *27*(1), 2-23.

Karatzoglou, A., Amatriain, X., Baltrunas, L., & Oliver, N. (2010). Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth acm conference on recommender systems* (pp. 79–86). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/1864708.1864727` doi: 10.1145/1864708.1864727

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th acm sigkdd international conference on knowledge discovery and data mining* (pp. 426–434). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/1401890.1401944` doi: 10.1145/1401890.1401944

Koren, Y., Bell, R., & Volinsky, C. (2009, January). Matrix factorization techniques for recommender systems. *Computer*, *42*, 30-37. doi: 10.1109/MC.2009.263

Lathia, N., Hailes, S., & Capra, L. (2008, Mars). The effect of correlation coefficients on communities of recommenders. *SAC '08 Proceedings of the 2008 ACM symposium on Applied computing*, 2000-2005. doi: 10.1145/1363686.1364172

Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In T. Leen, T. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems 13* (pp. 556–562). MIT Press. Retrieved from `http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf`

Musto, C., Semeraro, G., & Lops, P. (2013, August). Contextual eVSM: A Content-Based Context-Aware Recommendation Framework Based on Distributional Semantics. *E-Commerce and Web Technologies, 14th International Conference*, *152*($10.1007/978-3-642-39878-0\_12$), 125-136.

Polytech. (2013, October-November 28-01). Context'13. *The Interdisciplinary Community of Context.* (`http://www.polytech.univ-savoie.fr/index.php?id=context-13-call-for-papers&L=0`)

Qian, G., Sural, S., Gu, Y., & Pramanik, S. (2004, Mars). Similarity between euclidean and cosine angle distance for nearest neighbor queries. *SAC '04 Proceedings of the 2004 ACM symposium on Applied computing*, 1232-1237. doi: 1-58113-812-1

Ranta, A., & Forsberg, M. (2012). *Implementing programming languages: an introduction to compilers and interpreters* (Vol. 16.). London: College Publications. Retrieved from `www.summon.com`

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. (2011). *Recommender systems handbook.* Springer.

Shlens, J. (2005). A tutorial on principal component analysis. *Systems Neurobiology Laboratory, University of California at San Diego*, *82*.

Slonim, N., Aharoni, E., & Crammer, K. (2013). Hartigan's k-means versus lloyd's k-means: Is it time for a change? In *Proceedings of the twenty-third international joint conference on artificial intelligence* (pp. 1677–1684). AAAI Press. Retrieved from `http://dl.acm.org/citation.cfm?id=2540128.2540369`

Sparr, G. (1975). *Linjär algebra.* Studentlitteratur, Lund.

Thompson, C. (2008, November 21). If you liked this, you're sure to love that. *The New York Times*. (`http://www.nytimes.com/2008/11/23/magazine/23Netflix-t.html?_r=0`)

Vattani, A. (2011). k-means requires exponentially many iterations even in the plane. *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, *45*(4), 596-616.

Wang, Y.-X., & Zhang, Y.-J. (2013, June). Nonnegative matrix factorization: A comprehensive review. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, *25*, 1336-1353. doi: 10.1109/TKDE.2012.51

Appendices

Appendix A

Category A

1	Training Period	2013-03-25 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
2	Training Period	2013-03-26 00:00 — 2013-04-01 10:00
	Testing Period	2013-04-01 10:00 — 2013-04-01 12:00
3	Training Period	2013-03-27 00:00 — 2013-04-02 10:00
	Testing Period	2013-04-02 10:00 — 2013-04-02 12:00
4	Training Period	2013-03-28 00:00 — 2013-04-03 10:00
	Testing Period	2013-04-03 10:00 — 2013-04-03 12:00
5	Training Period	2013-03-29 00:00 — 2013-04-04 10:00
	Testing Period	2013-04-04 10:00 — 2013-04-04 12:00
6	Training Period	2013-03-30 00:00 — 2013-04-05 10:00
	Testing Period	2013-04-05 10:00 — 2013-04-05 12:00
7	Training Period	2013-03-31 00:00 — 2013-04-06 10:00
	Testing Period	2013-04-06 10:00 — 2013-04-06 12:00
8	Training Period	2013-04-01 00:00 — 2013-04-07 10:00
	Testing Period	2013-04-07 10:00 — 2013-04-07 12:00

Category B		
1	Training Period	2013-03-25 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
2	Training Period	2013-03-26 05:00 — 2013-04-01 15:00
	Testing Period	2013-04-01 15:00 — 2013-04-01 17:00
3	Training Period	2013-03-27 05:00 — 2013-04-02 15:00
	Testing Period	2013-04-02 15:00 — 2013-04-02 17:00
4	Training Period	2013-03-28 05:00 — 2013-04-03 15:00
	Testing Period	2013-04-03 15:00 — 2013-04-03 17:00
5	Training Period	2013-03-29 05:00 — 2013-04-04 15:00
	Testing Period	2013-04-04 15:00 — 2013-04-04 17:00
6	Training Period	2013-03-30 05:00 — 2013-04-05 15:00
	Testing Period	2013-04-05 15:00 — 2013-04-05 17:00
7	Training Period	2013-03-31 05:00 — 2013-04-06 15:00
	Testing Period	2013-04-06 15:00 — 2013-04-06 17:00
8	Training Period	2013-04-01 05:00 — 2013-04-07 15:00
	Testing Period	2013-04-07 15:00 — 2013-04-07 17:00

Category C		
1	Training Period	2013-03-25 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
2	Training Period	2013-03-26 09:00 — 2013-04-01 19:00
	Testing Period	2013-04-01 19:00 — 2013-04-01 21:00
3	Training Period	2013-03-27 09:00 — 2013-04-02 19:00
	Testing Period	2013-04-02 19:00 — 2013-04-02 21:00
4	Training Period	2013-03-28 09:00 — 2013-04-03 19:00
	Testing Period	2013-04-03 19:00 — 2013-04-03 21:00
5	Training Period	2013-03-29 09:00 — 2013-04-04 19:00
	Testing Period	2013-04-04 19:00 — 2013-04-04 21:00
6	Training Period	2013-03-30 09:00 — 2013-04-05 19:00
	Testing Period	2013-04-05 19:00 — 2013-04-05 21:00
7	Training Period	2013-03-31 09:00 — 2013-04-06 19:00
	Testing Period	2013-04-06 19:00 — 2013-04-06 21:00
8	Training Period	2013-04-01 09:00 — 2013-04-07 19:00
	Testing Period	2013-04-07 19:00 — 2013-04-07 21:00

Category D		
1	Training Period	2013-03-25 00:00 — 2013-03-31 24:00
	Testing Period	-||-
2	Training Period	2013-03-26 00:00 — 2013-04-01 24:00
	Testing Period	-||-
3	Training Period	2013-03-27 00:00 — 2013-04-02 24:00
	Testing Period	-||-
4	Training Period	2013-03-28 00:00 — 2013-04-03 24:00
	Testing Period	-||-
5	Training Period	2013-03-29 00:00 — 2013-04-04 24:00
	Testing Period	-||-
6	Training Period	2013-03-30 00:00 — 2013-04-05 24:00
	Testing Period	-||-
7	Training Period	2013-04-31 00:00 — 2013-04-06 24:00
	Testing Period	-||-

Category E		
1	Training Period	2012-10-01 00:00 — 2012-10-08 10:00
	Testing Period	2012-10-08 10:00 — 2012-10-08 12:00
2	Training Period	2012-10-02 00:00 — 2012-10-09 10:00
	Testing Period	2012-10-09 10:00 — 2012-10-09 12:00
3	Training Period	2012-10-03 00:00 — 2012-10-10 10:00
	Testing Period	2012-10-10 10:00 — 2012-10-10 12:00
4	Training Period	2012-10-04 00:00 — 2012-10-11 10:00
	Testing Period	2012-10-11 10:00 — 2012-10-11 12:00
5	Training Period	2012-10-05 00:00 — 2012-10-12 10:00
	Testing Period	2012-10-12 10:00 — 2012-10-12 12:00
6	Training Period	2012-10-06 00:00 — 2012-10-13 10:00
	Testing Period	2012-10-13 10:00 — 2012-10-13 12:00
7	Training Period	2012-10-07 00:00 — 2012-10-14 10:00
	Testing Period	2012-10-14 10:00 — 2012-10-14 12:00
8	Training Period	2012-10-08 00:00 — 2012-10-15 10:00
	Testing Period	2012-10-15 10:00 — 2012-10-15 12:00

Category F

1	Training Period	2012-10-01 05:00 — 2012-10-08 15:00
	Testing Period	2012-10-08 15:00 — 2012-10-08 17:00
2	Training Period	2012-10-02 05:00 — 2012-10-09 15:00
	Testing Period	2012-10-09 15:00 — 2012-10-09 17:00
3	Training Period	2012-10-03 05:00 — 2012-10-10 15:00
	Testing Period	2012-10-10 15:00 — 2012-10-10 17:00
4	Training Period	2012-10-04 05:00 — 2012-10-11 15:00
	Testing Period	2012-10-11 15:00 — 2012-10-11 17:00
5	Training Period	2012-10-05 05:00 — 2012-10-12 15:00
	Testing Period	2012-10-12 15:00 — 2012-10-12 17:00
6	Training Period	2012-10-06 05:00 — 2012-10-13 15:00
	Testing Period	2012-10-13 15:00 — 2012-10-13 17:00
7	Training Period	2012-10-07 05:00 — 2012-10-14 15:00
	Testing Period	2012-10-14 15:00 — 2012-10-14 17:00
8	Training Period	2012-10-08 05:00 — 2012-10-15 15:00
	Testing Period	2012-10-15 15:00 — 2012-10-15 17:00

Category G		
1	Training Period	2012-10-01 09:00 — 2012-10-08 19:00
	Testing Period	2012-10-08 19:00 — 2012-10-08 21:00
2	Training Period	2012-10-02 09:00 — 2012-10-09 19:00
	Testing Period	2012-10-09 19:00 — 2012-10-09 21:00
3	Training Period	2012-10-03 09:00 — 2012-10-10 19:00
	Testing Period	2012-10-10 19:00 — 2012-10-10 21:00
4	Training Period	2012-10-04 09:00 — 2012-10-11 19:00
	Testing Period	2012-10-11 19:00 — 2012-10-11 21:00
5	Training Period	2012-10-05 09:00 — 2012-10-12 19:00
	Testing Period	2012-10-12 19:00 — 2012-10-12 21:00
6	Training Period	2012-10-06 09:00 — 2012-10-13 19:00
	Testing Period	2012-10-13 19:00 — 2012-10-13 21:00
7	Training Period	2012-10-07 09:00 — 2012-10-14 19:00
	Testing Period	2012-10-14 19:00 — 2012-10-14 21:00
8	Training Period	2012-10-08 09:00 — 2012-10-15 19:00
	Testing Period	2012-10-15 19:00 — 2012-10-15 21:00

Category H		
1	Training Period	2012-11-01 00:00 — 2012-11-08 10:00
	Testing Period	2012-11-08 10:00 — 2012-11-08 12:00
2	Training Period	2012-11-02 00:00 — 2012-11-09 10:00
	Testing Period	2012-11-09 10:00 — 2012-11-09 12:00
3	Training Period	2012-11-03 00:00 — 2012-11-10 10:00
	Testing Period	2012-11-10 10:00 — 2012-11-10 12:00
4	Training Period	2012-11-04 00:00 — 2012-11-11 10:00
	Testing Period	2012-11-11 10:00 — 2012-11-11 12:00
5	Training Period	2012-11-05 00:00 — 2012-11-12 10:00
	Testing Period	2012-11-12 10:00 — 2012-11-12 12:00
6	Training Period	2012-11-06 00:00 — 2012-11-13 10:00
	Testing Period	2012-11-13 10:00 — 2012-11-13 12:00
7	Training Period	2012-11-07 00:00 — 2012-11-14 10:00
	Testing Period	2012-11-14 10:00 — 2012-11-14 12:00
8	Training Period	2012-11-08 00:00 — 2012-11-15 10:00
	Testing Period	2012-11-15 10:00 — 2012-11-15 12:00

Category I		
1	Training Period	2012-11-01 05:00 — 2012-11-08 15:00
	Testing Period	2012-11-08 15:00 — 2012-11-08 17:00
2	Training Period	2012-11-02 05:00 — 2012-11-09 15:00
	Testing Period	2012-11-09 15:00 — 2012-11-09 17:00
3	Training Period	2012-11-03 05:00 — 2012-11-10 15:00
	Testing Period	2012-11-10 15:00 — 2012-11-10 17:00
4	Training Period	2012-11-04 05:00 — 2012-11-11 15:00
	Testing Period	2012-11-11 15:00 — 2012-11-11 17:00
5	Training Period	2012-11-05 05:00 — 2012-11-12 15:00
	Testing Period	2012-11-12 15:00 — 2012-11-12 17:00
6	Training Period	2012-11-06 05:00 — 2012-11-13 15:00
	Testing Period	2012-11-13 15:00 — 2012-11-13 17:00
7	Training Period	2012-11-07 05:00 — 2012-11-14 15:00
	Testing Period	2012-11-14 15:00 — 2012-11-14 17:00
8	Training Period	2012-11-08 05:00 — 2012-11-15 15:00
	Testing Period	2012-11-15 15:00 — 2012-11-15 17:00

Category J		
1	Training Period	2012-11-01 09:00 — 2012-11-08 19:00
	Testing Period	2012-11-08 19:00 — 2012-11-08 21:00
2	Training Period	2012-11-02 09:00 — 2012-11-09 19:00
	Testing Period	2012-11-09 19:00 — 2012-11-09 21:00
3	Training Period	2012-11-03 09:00 — 2012-11-10 19:00
	Testing Period	2012-11-10 19:00 — 2012-11-10 21:00
4	Training Period	2012-11-04 09:00 — 2012-11-11 19:00
	Testing Period	2012-11-11 19:00 — 2012-11-11 21:00
5	Training Period	2012-11-05 09:00 — 2012-11-12 19:00
	Testing Period	2012-11-12 19:00 — 2012-11-12 21:00
6	Training Period	2012-11-06 09:00 — 2012-11-13 19:00
	Testing Period	2012-11-13 19:00 — 2012-11-13 21:00
7	Training Period	2012-11-07 09:00 — 2012-11-14 19:00
	Testing Period	2012-11-14 19:00 — 2012-11-14 21:00
8	Training Period	2012-11-08 09:00 — 2012-11-15 19:00
	Testing Period	2012-11-15 19:00 — 2012-11-15 21:00

Category K		
1	Training Period	2012-12-01 00:00 — 2012-12-08 10:00
	Testing Period	2012-12-08 10:00 — 2012-12-08 12:00
2	Training Period	2012-12-02 00:00 — 2012-12-09 10:00
	Testing Period	2012-12-09 10:00 — 2012-12-09 12:00
3	Training Period	2012-12-03 00:00 — 2012-12-10 10:00
	Testing Period	2012-12-10 10:00 — 2012-12-10 12:00
4	Training Period	2012-12-04 00:00 — 2012-12-11 10:00
	Testing Period	2012-12-11 10:00 — 2012-12-11 12:00
5	Training Period	2012-12-05 00:00 — 2012-12-12 10:00
	Testing Period	2012-12-12 10:00 — 2012-12-12 12:00
6	Training Period	2012-12-06 00:00 — 2012-12-13 10:00
	Testing Period	2012-12-13 10:00 — 2012-12-13 12:00
7	Training Period	2012-12-07 00:00 — 2012-12-14 10:00
	Testing Period	2012-12-14 10:00 — 2012-12-14 12:00
8	Training Period	2012-12-08 00:00 — 2012-12-15 10:00
	Testing Period	2012-12-15 10:00 — 2012-12-15 12:00

Category L		
1	Training Period	2012-12-01 05:00 — 2012-12-08 15:00
	Testing Period	2012-12-08 15:00 — 2012-12-08 17:00
2	Training Period	2012-12-02 05:00 — 2012-12-09 15:00
	Testing Period	2012-12-09 15:00 — 2012-12-09 17:00
3	Training Period	2012-12-03 05:00 — 2012-12-10 15:00
	Testing Period	2012-12-10 15:00 — 2012-12-10 17:00
4	Training Period	2012-12-04 05:00 — 2012-12-11 15:00
	Testing Period	2012-12-11 15:00 — 2012-12-11 17:00
5	Training Period	2012-12-05 05:00 — 2012-12-12 15:00
	Testing Period	2012-12-12 15:00 — 2012-12-12 17:00
6	Training Period	2012-12-06 05:00 — 2012-12-13 15:00
	Testing Period	2012-12-13 15:00 — 2012-12-13 17:00
7	Training Period	2012-12-07 05:00 — 2012-12-14 15:00
	Testing Period	2012-12-14 15:00 — 2012-12-14 17:00
8	Training Period	2012-12-08 05:00 — 2012-12-15 15:00
	Testing Period	2012-12-15 15:00 — 2012-12-15 17:00

Category M

1	Training Period	2012-12-01 09:00 — 2012-12-08 19:00
	Testing Period	2012-12-08 19:00 — 2012-12-08 21:00
2	Training Period	2012-12-02 09:00 — 2012-12-09 19:00
	Testing Period	2012-12-09 19:00 — 2012-12-09 21:00
3	Training Period	2012-12-03 09:00 — 2012-12-10 19:00
	Testing Period	2012-12-10 19:00 — 2012-12-10 21:00
4	Training Period	2012-12-04 09:00 — 2012-12-11 19:00
	Testing Period	2012-12-11 19:00 — 2012-12-11 21:00
5	Training Period	2012-12-05 09:00 — 2012-12-12 19:00
	Testing Period	2012-12-12 19:00 — 2012-12-12 21:00
6	Training Period	2012-12-06 09:00 — 2012-12-13 19:00
	Testing Period	2012-12-13 19:00 — 2012-12-13 21:00
7	Training Period	2012-12-07 09:00 — 2012-12-14 19:00
	Testing Period	2012-12-14 19:00 — 2012-12-14 21:00
8	Training Period	2012-12-08 09:00 — 2012-12-15 19:00
	Testing Period	2012-12-15 19:00 — 2012-12-15 21:00

Category N		
1	Training Period	2013-03-17 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
2	Training Period	2013-03-18 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
3	Training Period	2013-03-19 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
4	Training Period	2013-03-20 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
5	Training Period	2013-03-21 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
6	Training Period	2013-03-22 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
7	Training Period	2013-03-23 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
8	Training Period	2013-03-24 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
9	Training Period	2013-03-25 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
10	Training Period	2013-03-26 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
11	Training Period	2013-03-27 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
12	Training Period	2013-03-28 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
13	Training Period	2013-03-29 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00
14	Training Period	2013-03-30 00:00 — 2013-03-31 10:00
	Testing Period	2013-03-31 10:00 — 2013-03-31 12:00

Category O		
1	Training Period	2013-03-17 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
2	Training Period	2013-03-18 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
3	Training Period	2013-03-19 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
4	Training Period	2013-03-20 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
5	Training Period	2013-03-21 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
6	Training Period	2013-03-22 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
7	Training Period	2013-03-23 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
8	Training Period	2013-03-24 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
9	Training Period	2013-03-25 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
10	Training Period	2013-03-26 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
11	Training Period	2013-03-27 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
12	Training Period	2013-03-28 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
13	Training Period	2013-03-29 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00
14	Training Period	2013-03-30 05:00 — 2013-03-31 15:00
	Testing Period	2013-03-31 15:00 — 2013-03-31 17:00

Category P		
1	Training Period	2013-03-17 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
2	Training Period	2013-03-18 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
3	Training Period	2013-03-19 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
4	Training Period	2013-03-20 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
5	Training Period	2013-03-21 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
6	Training Period	2013-03-22 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
7	Training Period	2013-03-23 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
8	Training Period	2013-03-24 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
9	Training Period	2013-03-25 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
10	Training Period	2013-03-26 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
11	Training Period	2013-03-27 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
12	Training Period	2013-03-28 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
13	Training Period	2013-03-29 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00
14	Training Period	2013-03-30 09:00 — 2013-03-31 19:00
	Testing Period	2013-03-31 19:00 — 2013-03-31 21:00

Category Q

1	Training Period	2012-09-01 00:00 — 2012-09-07 10:00
	Testing Period	2012-09-07 10:00 — 2012-09-07 12:00
2	Training Period	2012-10-01 00:00 — 2012-10-07 10:00
	Testing Period	2012-10-07 10:00 — 2012-10-07 12:00
3	Training Period	2012-11-01 00:00 — 2012-11-07 10:00
	Testing Period	2012-11-07 10:00 — 2012-11-07 12:00
4	Training Period	2012-12-01 00:00 — 2012-12-07 10:00
	Testing Period	2012-12-07 10:00 — 2012-12-07 12:00
5	Training Period	2013-01-01 00:00 — 2013-01-07 10:00
	Testing Period	2013-01-07 10:00 — 2013-01-07 12:00
6	Training Period	2013-02-01 00:00 — 2013-02-07 10:00
	Testing Period	2013-02-07 10:00 — 2013-02-07 12:00
7	Training Period	2013-03-01 00:00 — 2013-03-07 10:00
	Testing Period	2013-03-07 10:00 — 2013-03-07 12:00
8	Training Period	2013-04-01 00:00 — 2013-04-07 10:00
	Testing Period	2013-04-07 10:00 — 2013-04-07 12:00
9	Training Period	2013-05-01 00:00 — 2013-05-07 10:00
	Testing Period	2013-05-07 10:00 — 2013-05-07 12:00

Category R		
1	Training Period	2012-09-01 05:00 — 2012-09-07 15:00
	Testing Period	2012-09-07 15:00 — 2012-09-07 17:00
2	Training Period	2012-10-01 05:00 — 2012-10-07 15:00
	Testing Period	2012-10-07 15:00 — 2012-10-07 17:00
3	Training Period	2012-11-01 05:00 — 2012-11-07 15:00
	Testing Period	2012-11-07 15:00 — 2012-11-07 17:00
4	Training Period	2012-12-01 05:00 — 2012-12-07 15:00
	Testing Period	2012-12-07 15:00 — 2012-12-07 17:00
5	Training Period	2013-01-01 05:00 — 2013-01-07 15:00
	Testing Period	2013-01-07 15:00 — 2013-01-07 17:00
6	Training Period	2013-02-01 05:00 — 2013-02-07 15:00
	Testing Period	2013-02-07 15:00 — 2013-02-07 17:00
7	Training Period	2013-03-01 05:00 — 2013-03-07 15:00
	Testing Period	2013-03-07 15:00 — 2013-03-07 17:00
8	Training Period	2013-04-01 05:00 — 2013-04-07 15:00
	Testing Period	2013-04-07 15:00 — 2013-04-07 17:00
9	Training Period	2013-05-01 05:00 — 2013-05-07 15:00
	Testing Period	2013-05-07 15:00 — 2013-05-07 17:00

Category S		
1	Training Period	2012-09-01 09:00 — 2012-09-07 19:00
	Testing Period	2012-09-07 19:00 — 2012-09-07 21:00
2	Training Period	2012-10-01 09:00 — 2012-10-07 19:00
	Testing Period	2012-10-07 19:00 — 2012-10-07 21:00
3	Training Period	2012-11-01 09:00 — 2012-11-07 19:00
	Testing Period	2012-11-07 19:00 — 2012-11-07 21:00
4	Training Period	2012-12-01 09:00 — 2012-12-07 19:00
	Testing Period	2012-12-07 19:00 — 2012-12-07 21:00
5	Training Period	2013-01-01 09:00 — 2013-01-07 19:00
	Testing Period	2013-01-07 19:00 — 2013-01-07 21:00
6	Training Period	2013-02-01 09:00 — 2013-02-07 19:00
	Testing Period	2013-02-07 19:00 — 2013-02-07 21:00
7	Training Period	2013-03-01 09:00 — 2013-03-07 19:00
	Testing Period	2013-03-07 19:00 — 2013-03-07 21:00
8	Training Period	2013-04-01 09:00 — 2013-04-07 19:00
	Testing Period	2013-04-07 19:00 — 2013-04-07 21:00
9	Training Period	2013-05-01 09:00 — 2013-05-07 19:00
	Testing Period	2013-05-07 19:00 — 2013-05-07 21:00

Category T		
1	Training Period	2013-03-02 10:00 — 2013-03-02 12:00
	Training Period	2013-03-09 10:00 — 2013-03-09 12:00
	Training Period	2013-03-16 10:00 — 2013-03-16 12:00
	Training Period	2013-03-23 10:00 — 2013-03-23 12:00
	Training Period	2013-03-30 10:00 — 2013-03-30 12:00
Test	Testing Period	2013-04-06 10:00 — 2013-04-06 12:00
2	Training Period	2013-03-02 15:00 — 2013-03-02 17:00
	Training Period	2013-03-09 15:00 — 2013-03-09 17:00
	Training Period	2013-03-16 15:00 — 2013-03-16 17:00
	Training Period	2013-03-23 15:00 — 2013-03-23 17:00
	Training Period	2013-03-30 15:00 — 2013-03-30 17:00
Test	Testing Period	2013-04-06 15:00 — 2013-04-06 17:00
3	Training Period	2013-03-02 19:00 — 2013-03-02 21:00
	Training Period	2013-03-09 19:00 — 2013-03-09 21:00
	Training Period	2013-03-16 19:00 — 2013-03-16 21:00
	Training Period	2013-03-23 19:00 — 2013-03-23 21:00
	Training Period	2013-03-30 19:00 — 2013-03-30 21:00
Test	Testing Period	2013-04-06 19:00 — 2013-04-06 21:00

Category U		
1	Training Period	2013-03-03 10:00 — 2013-03-03 12:00
	Training Period	2013-03-10 10:00 — 2013-03-10 12:00
	Training Period	2013-03-17 10:00 — 2013-03-17 12:00
	Training Period	2013-03-24 10:00 — 2013-03-24 12:00
	Training Period	2013-03-31 10:00 — 2013-03-31 12:00
Test	Testing Period	2013-04-07 10:00 — 2013-04-07 12:00
2	Training Period	2013-03-03 15:00 — 2013-03-03 17:00
	Training Period	2013-03-10 15:00 — 2013-03-10 17:00
	Training Period	2013-03-17 15:00 — 2013-03-17 17:00
	Training Period	2013-03-24 15:00 — 2013-03-24 17:00
	Training Period	2013-03-31 15:00 — 2013-03-31 17:00
Test	Testing Period	2013-04-07 15:00 — 2013-04-07 17:00
3	Training Period	2013-03-03 19:00 — 2013-03-03 21:00
	Training Period	2013-03-10 19:00 — 2013-03-10 21:00
	Training Period	2013-03-17 19:00 — 2013-03-17 21:00
	Training Period	2013-03-24 19:00 — 2013-03-24 21:00
	Training Period	2013-03-31 19:00 — 2013-03-31 21:00
Test	Testing Period	2013-04-07 19:00 — 2013-04-07 21:00

Category V (Training/testing only on weekdays, not on weekends)		
1	Training Period	2012-11-01 00:00 — 2012-11-19 10:00
	Testing Period	2012-11-22 00:00 — 2012-11-22 10:00
2	Training Period	2012-11-02 00:00 — 2012-11-22 10:00
	Testing Period	2012-11-23 00:00 — 2012-11-23 10:00
3	Training Period	2012-11-03 00:00 — 2012-11-23 10:00
	Testing Period	2012-11-24 00:00 — 2012-11-24 10:00
4	Training Period	2012-11-04 00:00 — 2012-11-24 10:00
	Testing Period	2012-11-25 00:00 — 2012-11-25 10:00
5	Training Period	2012-11-05 00:00 — 2012-11-25 10:00
	Testing Period	2012-11-26 00:00 — 2012-11-26 10:00
6	Training Period	2012-11-08 00:00 — 2012-11-26 10:00
	Testing Period	2012-11-29 00:00 — 2012-11-29 10:00
7	Training Period	2012-11-09 00:00 — 2012-11-29 10:00
	Testing Period	2012-11-30 00:00 — 2012-11-30 10:00
8	Training Period	2012-11-10 00:00 — 2012-11-30 10:00
	Testing Period	2012-11-31 00:00 — 2012-11-31 10:00

Category W (Training/testing only on weekdays, not on weekends)		
1	Training Period	2012-11-01 10:00 — 2012-11-19 17:00
	Testing Period	2012-11-22 10:00 — 2012-11-22 17:00
2	Training Period	2012-11-02 10:00 — 2012-11-22 17:00
	Testing Period	2012-11-23 10:00 — 2012-11-23 17:00
3	Training Period	2012-11-03 10:00 — 2012-11-23 17:00
	Testing Period	2012-11-24 10:00 — 2012-11-24 17:00
4	Training Period	2012-11-04 10:00 — 2012-11-24 17:00
	Testing Period	2012-11-25 10:00 — 2012-11-25 17:00
5	Training Period	2012-11-05 10:00 — 2012-11-25 17:00
	Testing Period	2012-11-26 10:00 — 2012-11-26 17:00
6	Training Period	2012-11-08 10:00 — 2012-11-26 17:00
	Testing Period	2012-11-29 10:00 — 2012-11-29 17:00
7	Training Period	2012-11-09 10:00 — 2012-11-29 17:00
	Testing Period	2012-11-30 10:00 — 2012-11-30 17:00
8	Training Period	2012-11-10 10:00 — 2012-11-30 17:00
	Testing Period	2012-11-31 10:00 — 2012-11-31 17:00

Category X (Training/testing only on weekdays, not on weekends)		
1	Training Period	2012-11-01 17:00 — 2012-11-19 00:00
	Testing Period	2012-11-22 17:00 — 2012-11-23 00:00
2	Training Period	2012-11-02 17:00 — 2012-11-22 00:00
	Testing Period	2012-11-23 17:00 — 2012-11-24 00:00
3	Training Period	2012-11-03 17:00 — 2012-11-23 00:00
	Testing Period	2012-11-24 17:00 — 2012-11-25 00:00
4	Training Period	2012-11-04 17:00 — 2012-11-24 00:00
	Testing Period	2012-11-25 17:00 — 2012-11-26 00:00
5	Training Period	2012-11-05 17:00 — 2012-11-25 00:00
	Testing Period	2012-11-26 17:00 — 2012-11-27 00:00
6	Training Period	2012-11-08 17:00 — 2012-11-26 00:00
	Testing Period	2012-11-29 17:00 — 2012-11-30 00:00
7	Training Period	2012-11-09 17:00 — 2012-11-29 00:00
	Testing Period	2012-11-30 17:00 — 2012-11-31 00:00
8	Training Period	2012-11-10 17:00 — 2012-11-30 00:00
	Testing Period	2012-11-31 17:00 — 2012-12-01 00:00

Category Y		
1	Training Period	2013-01-01 00:00 — 2013-03-31 00:00
	Testing Period	2013-03-31 00:00 — 2013-04-01 00:00
2	Training Period	2013-03-01 00:00 — 2013-03-31 00:00
	Testing Period	2013-03-31 00:00 — 2013-04-01 00:00
3	Training Period	2013-03-24 00:00 — 2013-03-31 00:00
	Testing Period	2013-03-31 00:00 — 2013-04-01 00:00

Appendix B

Cluster test

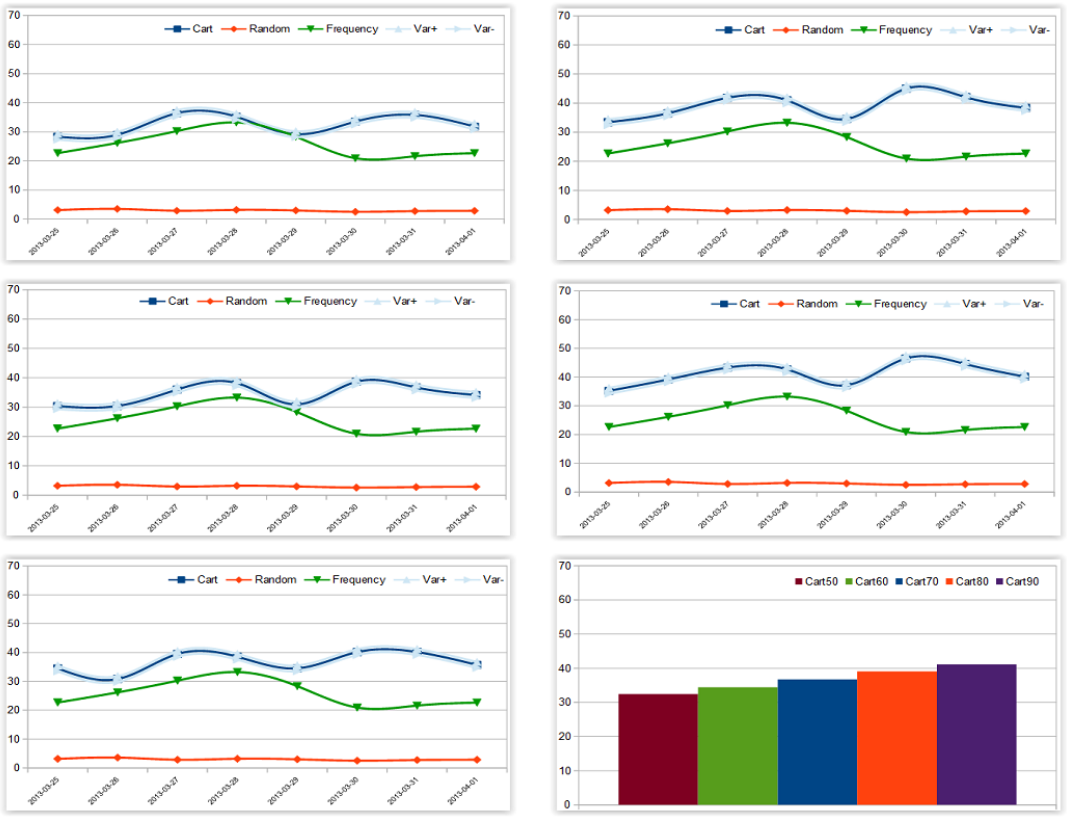


Table 1: The graphs show how the cluster size changes the performance on the average of catagories A,B and C.