Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

Bachelor's Thesis

Kasper Halme

TREND-BASED RECOMMENDATION MODEL

Examiners: D.Sc. (Tech.) Ari Happonen, PhD student Antti Herala

Supervisor: D.Sc. (Tech.) Ari Happonen, D.Sc. (Tech.) Jun Miyazaki

ABSTRACT

Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

Kasper Halme

Trend-based Recommendation model

Bachelor's Thesis

2016

26 pages, 4 figures, 3 tables.

Examiners: D.Sc. (Tech.) Ari Happonen, PhD student Antti Herala

Keywords: information retrieval, recommendation models, tensor analytics, gibbs sampling

As the amount of data to be handled increases, so does the need for new ways to analyze and parse it for relevant information. Despite this, many of the general recommendation models, such as Collaborative Filtering, only take advantage of the general characteristics giving only superficial information as a result. The aim of this bachelor's thesis is to find out what kind of results arises when the desired information is based on the found trends within data. These trends can be discovered by using a tensor analytics with Gibbs Motif Finding to provide an estimation of people's behavioral relations, which are then ranked according to their scores. In order to clarify the study, open electoral data was utilized for Trend-based and general recommendation model to compare their results. The obtained results differed significantly from each other. As a new phenomenon, Trend-based recommendation model ranked results with unique characteristics higher, when the general recommendation models results were homogeneous. In the light of given criterias the Trend-based recommendation model were assessed to be most accurate of the two.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
School of Business and Management
Tietotekniikan koulutusohjelma

Kasper Halme

Trendipohjainen suosittelumalli

Kandidaatintyö

2016

26 sivua, 4 kuvaa, 3 taulukkoa.

Tarkastajat: TkT Ari Happonen, nuorempi tutkija Antti Herala

Hakusanat: tiedonhaku, suosittelumallit, tensorianalytiikka, gibbs sampling

Käsiteltävissä olevan tiedon määrän kasvaessa tarvitaan uusia tapoja analysoida tietoa ja seuloa tietomassoista merkityksellistä informaatiota. Tästä huolimatta monet alan yleiset suosittelumallit, kuten Collaborative Filtering, hyödyntävät tietomassan yleisiä ominaisuuksija tarjoavat vain pinnallista informaatiota.

Tämän kandidaatintyön tavoitteena on selvittää, millaisia tuloksia saadaan, kun haluttu informaatio perustuu tietomassasta löydettyihin trendeihin. Nämä trendit saadaan selville käyttämällä tensorianalytiikkaa Gibbsin Motif Finding -metodilla ja tuottamalla arvio ihmisten käyttäytymismallien suhteista, jotka lopuksi sijoitetaan saadun arvon avulla järjestykseen. Tutkimuksen selventämiseksi Ylen avointa vaalidataa hyödynnettiin tietomassana trendipohjaisen ja yleisen suosittelumallin tulosten vertailua varten. Vertailusta saadut tulokset erottuivat toisistaan huomattavasti. Trendipohjainen malli sijoitti arvoiltaan vaihtelevia tuloksia tärkeämmiksi, kun yleisen mallin tulokset sen sijaan olivat arvoiltaan samanlaisia. Trendipohjainen suosittelumalli arvioitiin tarkemmaksi kun Collaborative Filteringin.

CONTENTS

1 INTRODUCTION

1.1 Background

A growing number of companies are trying to follow the example of technological trailblazers such as Amazon and Google. Currently both are effectively utilizing predictive modelling, forming their core competences from it. Amazon providing product recommendations and Google website search results. The two are well known to be leaders in their own respective areas [1] [2].

Alongside them, predictive modelling has become highly regarded due its usage with aggregated big data. Nobody really knows how to use this data, but there is a high chance that it might prelude the next big thing. Amazon and Google are already applying recommendations to forecast and personalize user news feed, to categorize their photos or to auto-complete sentences[3] [4]. These services are a part of their shared vision of; a personalized ecosystem of information for each user and client. This can be visibly seen from their recent efforts in both research and company acquisitions, Figure 1. The mentioned services, among the likes of Amazon's product recommender, aspire to this vision by trying to assure accurate search results. Their recommendation algorithms are ever evolving, but are usually based on ranked predictions using methods such as information retrieval and pattern recognition in an attempt to find similarities from your recent searches [5].

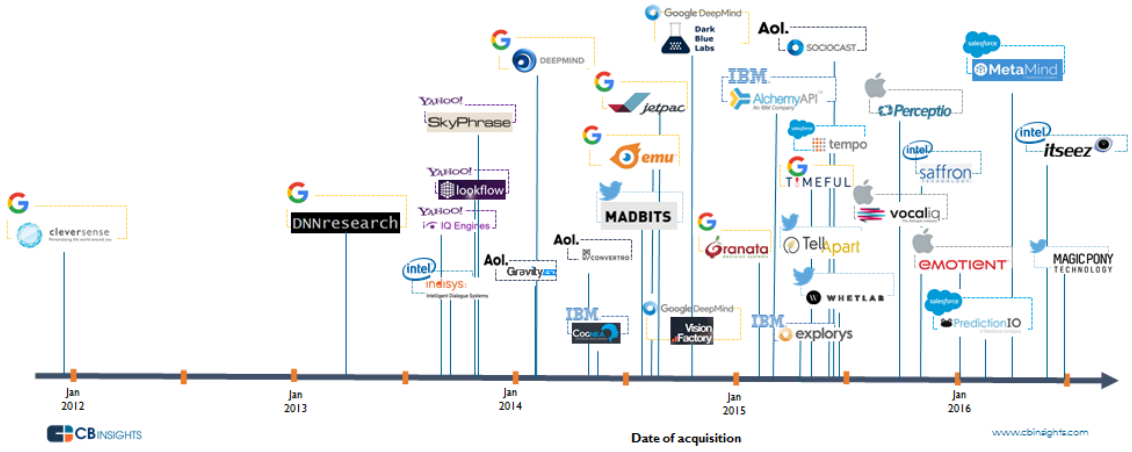


Figure 1. Acquisitions of predictive technologies.[6].

However, currently used recommendation algorithms have difficulties identifying how searched items are related to the user. Individuals often have a wide range of motivations when searching items unlike a plain item which usually belongs to relatively few genres. Without planning the result might be a complicated model imitating the item-user relations, followed by trouble which usually manifest itself as increased complexity. Often the models take a high computational toll in efficiency which limits how many times you can run the calculations.

The efficiency problem is solved by factoring and weighing the algorithms properties. It's computationally simpler to find repeatedly searched items from various users than it is to find similar users with the same type of relation to specific item. This means that the recommendations are produced without excessive computational cost by recommending popular items.

Unfortunately this solution is unlikely to provide a diverse set of results. Instead, factorization often produces the ”obvious” solution, lacking serendipity by not being affected by the motivation behind the searches. It also means that such results are not based on natural relations and looming trends, but instead on popular choices at each moment. This may become a critical flaw, for example when facing more specific challenges, such as “predicting trends” or “personalizing results” and marginalizing nuances, is no longer a viable option. Hence, a better solution is needed.

1.2 Motivation

Both the Enterprising and Intellectual communities have been speculating, how to better understand the relation between sets of personal data. The recent emerge of studies which combine tensors with recommendation systems have contributed a lot for this particular field. Using a tensor provides a chance to push vector data and receive a scalar [7]. This has the benefit of making a very complex data relation into a numerical value, proving its evident usefulness by becoming a widely researched subject. In addition the visual representation becomes simpler given the added dimension. But although many highly sophisticated methods have been created, none of them got very far without somehow influencing the results or complexity.

These issues point towards the core question: What kind of results emerges when recommendations are based on distinctive qualities of used personal data, and how they differ from regular recommendations?

To answer the question, this thesis will examine an alternate path; reducing the dominant problem of finding similar users by finding recurring trends amongst their behavior. This'll be accomplished with the combination of N^{th}-order Tensor and Motif Finding with Gibbs sampling to show a scalar value estimation of object relations between user preferences. These relations are afterwards scored by a recommender system technique called collaborative filtering which in a narrower sense, makes automatic predictions based on determined preferences [8] (Methods will be further elucidated in section 2). This methodological approach forms the base of Trend-based Recommendation Model creating a new viewpoint for further improvement. It will bring us one step closer to the personalized ecosystem of information by providing a method for recommendations, based on hidden trends from the data.

The experiments will use Finnish election candidate data which is gather from an election machine equipped with the answers from 1855 candidate. This Open data can be accessed as a CSV spreadsheet or through the website interface. CSV data table can be examined by any spreadsheet program. Election Data is published under a Creative Commons license which means that the election engine reply data may be used as part of any new works, as long as the source [9] is acknowledged. Responding to the electoral machine has already ended although after the release of the machine, some of the candidates had the discretion of extra time. Election Data also contains the names of the candidates who did not respond to the same questions but who nevertheless participated in the election.

1.3 Objectives and Restritions

The objective of this thesis is to use a Trend-based Recommendation Model as an alternate method, to provide recommendations, and to compare it to other models. The specific objectives are the following:

- First Objective is to combine N^{th}-order Tensor and Motif Finding with Gibbs sampling to efficiently create a series of recommendations based on looming trends.
- Second Objective is to score the recieved results using collaborative filtering so that a complete Trend-based Recommendation Model is produced.
- Third Objective is to use this Recommendation model to go through candidate data and provide a ranked recommendations for a voter, and compare the results to different models.

The specific assumptions and restrictions for Trend-based Recommendation Model are the following:

- Although feature augmentation is currently a fairly common methodology, its motif based variation Trend-based Recommendation Model has only been tested so far as a prototype and in this thesis to a selected data.
- The data format was changed for this experiment. Although it wasn't a necessity, simplyfying the data into boolean and integer values quickened the experimentation. This restriction narrowed the calculations and made the sophisticated computers obselite for this experiment.
- Overall any kind of data format could be pushed through the algorithm, but it should be noted that result are highly dependant in sheer quantity and quality of the used data.

1.4 Structure of the thesis

Section 2 gives the framework for the methodology and theory, related to the Trend-based Recommendation Model. Section 3 describes in detail the experimentation and the specific methods used for this thesis. Section 4 contains the results and further hypothesis. The results are discussed and the thesis is concluded in Section 5.

2 METHODOLOGY

2.1 Approach

The hypothesis is, that the computational complexity or scalability issues are averted by using an estimation of user shared patterns. The estimation, can be found by searching the N^{th}-order Tensor via Motif Finding with Gibbs sampling. After such patterns are found, one can approximate recommendations by using Collaborative filtering to give every object a score and to finalize the ranking.

Tensor method was chosen due to its competitiveness in unsupervised encapsulation of large-scale probabilistic patterns [10]. Multidimensional data such as time-series, can be modeled as tensors. It's an important data mining tool that can build on efficient linear algebraic libraries, and is more powerful and informative compared to matrix methods. However, tensor methods are not sample efficient, and they require a lot of samples to reach the preferred level of accuracy (assuming computation is not an issue).

Gibbs Sampling is another extremely useful technique for sampling multidimensional data distributions. The beauty of this model is, that it simplifies a complex high-dimensional problem by breaking it down into simple, low-dimensional problems. Its' useful when; sampling from the conditional distributions for each parameter (or blocks of them) is feasible [11]. It is most commonly used in the area of DNA/protein sequence analysis, but this thesis is trying to make it a part of a recommendation algorithm. The reason for this, can be traced back to the recent trends in which motif discovery algorithms were incorporated with additional information to improve the predictions accuracy. Essentially, Gibbs Sampling is mainly used to find a common sequence among protein sequences, but now with additional information from N-elements of N^{th}-order tensor, this method can effectively increase the signal/noise ratio, thus improving recommendations specificity and sensitivity. [12] In contrast a regular sequence data miner can also find a common sequences, but the remarkable difference of motif finding, is its better tolerance to a longer input sequence size, i.e. a better scalability. With this observation, the search space could be greatly reduced by changing sequence-based approach and take advantage of the potential for improvement. [12]

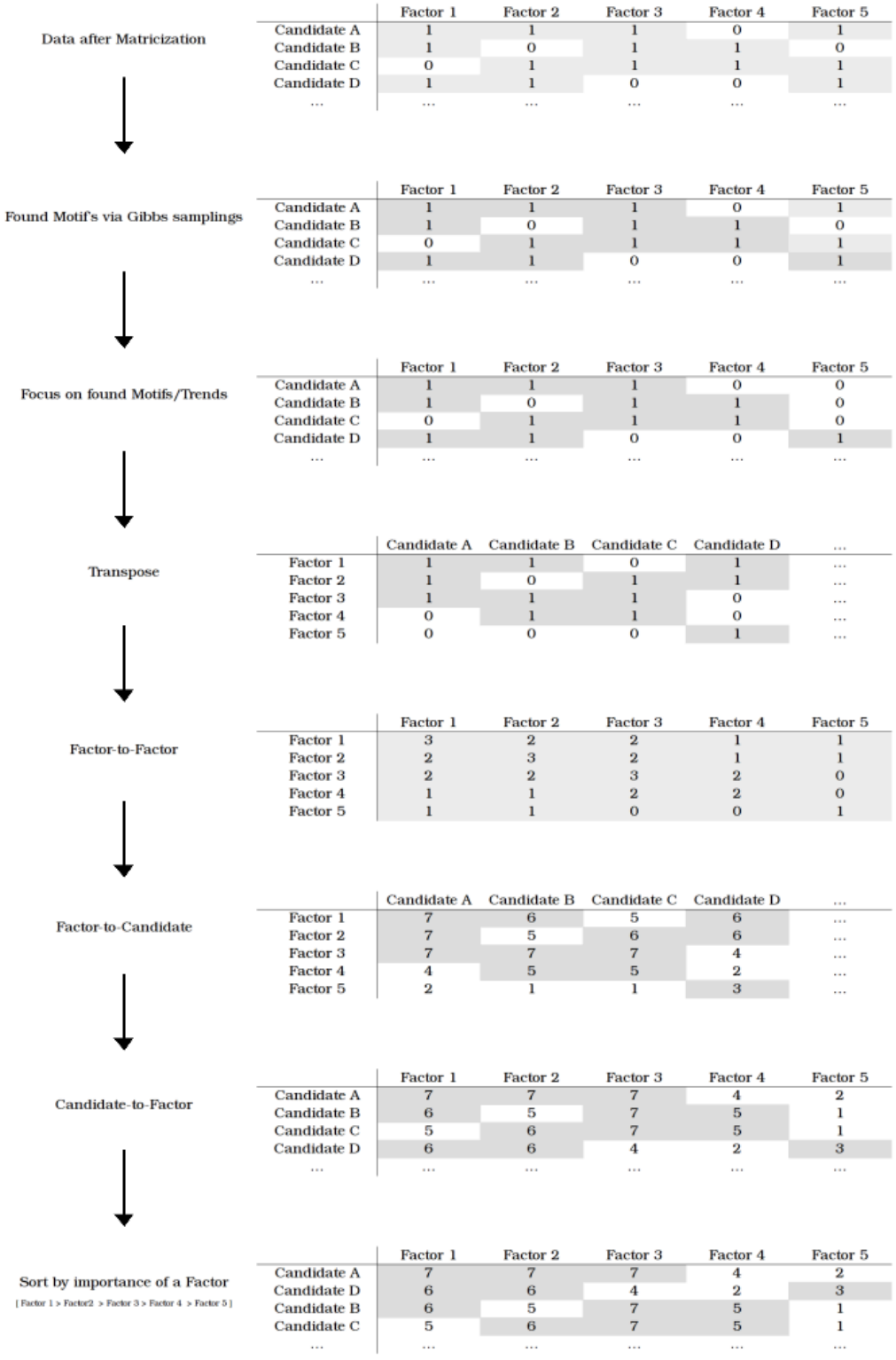
Collaborative filtering alleviates information overload by identifying which items a user will find worthwhile. This method was selected for its versatility and increased accuracy when handling large datasets. It can be adapted to almost any domain, and by focusing on identification of similar items across subspaces in the item space it escapes from the "filter bubble" problem [13].

Trend-based Recommendation Model is a hybrid recommender that combines two recommendation techniques to gain better performance with fewer drawbacks than of any individual one [14]. The chosen hybridization method is a feature augmentation, which employ is to produce a score for an item and incorporated the information into the processing of the next recommendation technique [14]. In trend-based Recommendation Model, Motif finding with Gibbs sampling produces the output used as an input feature for Modified Collaborative Filtering. Feature augmentation is attractive because it offers a way to improve the performance of a core system with fairly simple implementations. Additional functionality could be added by intermediaries who can use other techniques to augment the data itself. Note that this is different from feature combination in which raw data from different sources is combined [14].

Depending on the situation and the objective, a more accurate approximation for the final scoring can be achieved by changing the hybridization.

Table 1. Various options and methods for different situations, with each their own benefits.

Method	**Description**
Weighted	The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.
Switching	The system switches between recommendation techniques depending on the current situation.
Mixed	Recommendations from several different recommenders are presented at the same time.
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Cascade	One recommender refines the recommendations given by another.
Meta-level	The model learned by one recommender is used as input to another.



	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
Candidate A	1	1	1	0	1
Candidate B	1	0	1	1	0
Candidate C	0	1	1	1	1
Candidate D	1	1	0	0	1
...	...	...	...	...	...

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
Candidate A	1	1	1	0	1
Candidate B	1	0	1	1	0
Candidate C	0	1	1	1	1
Candidate D	1	1	0	0	1
...	...	...	...	...	...

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
Candidate A	1	1	1	0	0
Candidate B	1	0	1	1	0
Candidate C	0	1	1	1	0
Candidate D	1	1	0	0	1
...	...	...	...	...	...

	Candidate A	Candidate B	Candidate C	Candidate D	...
Factor 1	1	1	0	1	...
Factor 2	1	0	1	1	...
Factor 3	1	1	1	0	...
Factor 4	0	1	1	0	...
Factor 5	0	0	0	1	...

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
Factor 1	3	2	2	1	1
Factor 2	2	3	2	1	1
Factor 3	2	2	3	2	0
Factor 4	1	1	2	2	0
Factor 5	1	1	0	0	1

	Candidate A	Candidate B	Candidate C	Candidate D	...
Factor 1	7	6	5	6	...
Factor 2	7	5	6	6	...
Factor 3	7	7	7	4	...
Factor 4	4	5	5	2	...
Factor 5	2	1	1	3	...

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
Candidate A	7	7	7	4	2
Candidate B	6	5	7	5	1
Candidate C	5	6	7	5	1
Candidate D	6	6	4	2	3
...	...	...	...	...	...

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
Candidate A	7	7	7	4	2
Candidate D	6	6	4	2	3
Candidate B	6	5	7	5	1
Candidate C	5	6	7	5	1
...	...	...	...	...	...

Figure 2. Components and processing flow of trend-based recommendation

2.2 Theory

The field of research around recommendation systems is usually focused at recommending items based on user preferences, and it has become synonymous with collaborative filtering. This definition distinguishes recommendation systems from other systems that are more properly characterized as information retrieval or information filtering [15].

Even so, there are a wide variety of recommender systems available, although very few of them are based on natural relations. Therefore the greatest changes when comparing systems can be seen by using collaborative filtering as the frame of reference (many collaborative filtering systems rely on explicit statements of user opinion, such as ratings, to create recommendations) [15].

Trend-based Recommendation Model consists of three components:

1. Implementing the data to N^{th}-order Tensor
2. Applying Motif finding with Gibbs sampling to locate common patterns.
3. Use modified collaborative filtering to the patterns to provide ranking for recommendation.

The execution of these components requires the data to have the at least 3 parameters which are depicted in Section 2.2.1 as part of the N^{th}-order Tensor. Motif finding with Gibbs sampling is described in Section 2.2.2 and the Collaborative filtering in Section 2.2.3.

2.2.1 Tensor Model

N^{th}-order Tensor

A 3^{rd}-order tensor (or more formally known as an N^{th}-order) tensor is a multidimensional array. It's an element of the tensor product of N vector spaces, each of which has its own coordinate system to label a component of that array, N being the number of used components. For example; a first-order tensor is a vector, a second-order tensor is a matrix, and tensors of order three or higher are called higher-order tensors (Zero-order tensors are represented as single numbers or scalars). 3^{rd}-order tensor has three indices and is therefore a third-order tensor as shown in Figure 3 [16].

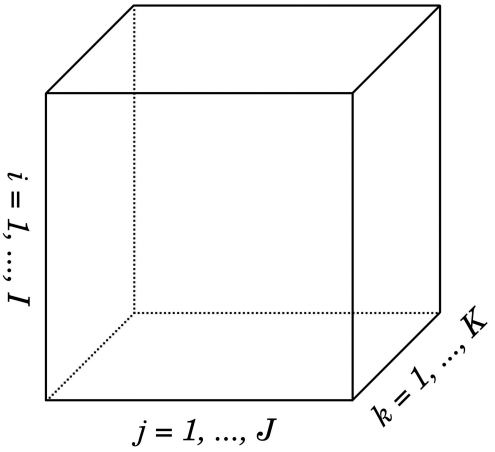


Figure 3. A third-order tensor $X \in \mathbb{R}^{I \times J \times K}$

The norm of a tensor $X \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the square root of the sum of the squares of all its elements, i.e .,

$$\| X \| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N}^2}.$$

This is analogous to the matrix Frobenius norm, which is denoted $\| A \|$ for a matrix A. The inner product of two same-sized tensors $X, Y \in \mathbb{R}^{I_1 \times I_2 \cdots I_N}$ is the sum of the products of their entries, i.e.,

$$\langle X, Y \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \dots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \dots i_N} y_{i_1 i_2 \dots i_N}.$$

It follows immediately that $\langle X, Y \rangle = \| X \|^2$ [16].

Matricization

Also known as unfolding or flattening, is the process of reordering the elements of an N^{th}-order array into a matrix. Some decomposition techniques apply matricization to tensors for extracting and explaining data properties in order to understand the data structure. Illustration of a matricization operation for a third-order tensor $X \in \mathbb{R}^{|U| \times |I| \times |T|}$ is given in Figure 4. The three modes/dimensions of the tensor X are users (U), items (I) and tags (T). Figure 4 shows the U-mode unfolding of the tensor X, denoted as $X_{(U)} \in \mathbb{R}^{|U| \times |I| \times |T|}$ [17].

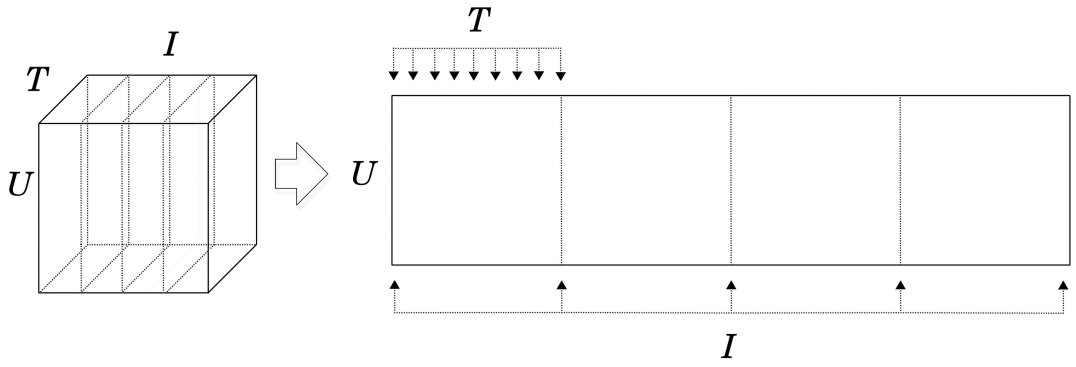


Figure 4. Matricization of a third-order tensor

Formally, in the mode-n matricization of a third-order tensor $X \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, a tensor element I_1, I_2, I_3 maps to a matrix element (i_n, j) [17], where

$$j = 1 + \sum_{k=1, k \neq n}^{3} (i_k - 1) J_k \text{ and } J_k = \prod_{m=1, m \neq n}^{k-1} I_m$$

2.2.2 Motif Finding with Gibbs Sampling

Monte Carlo Markov Chain

The goal of Bayesian inference is to maintain a full posterior probability distribution over a set of random variables. To maintain and use it, often involves computing integrals which are often intractable, except for most non-trivial models. Sampling algorithms based on Monte Carlo Markov Chain (MCMC) techniques are used as a solution for inference in such models [18].

MCMC techniques are often applied to solve integration and optimisation problems in large dimensional spaces especially in multidimentional tensors. These two types of problems are common e.g. in cosmology and astrophysics , which have directly benefited from MCMC (The models are often expensive to compute, there are many free parameters, and the observations are usually low in signal-to-noise) [19].

The basic logic of MCMC sampling is to estimate any desired expectation by ergodic averages. It's possible to compute any statistic of a posterior distribution as long as simulated samples (N) are available from the distribution:

$$E[f(s)]_P \approx \frac{1}{N} \sum_{i=1}^{N} f(s^{(i)})$$

where P is the posterior distribution of interest, $f(s)$ is the desired expectation, and $f(s^{(i)})$ is the i^{th} simulated sample from P. The mean can be estimated by $E[x]_P = \frac{1}{N} \sum_{i=1}^{N} x^{(i)}$ [18].

Gibbs sampling

Among the probabilistic methods Gibbs sampling is a MCMC technique suitable for obtaining samples from the posterior distribution. The basic idea is to split the distribution into blocks to generate posterior samples by sweeping through each block of variables separately, conditional on the most recent values of the other blocks.

Considering the random variables X_1, X_2, and X_3. These variables are set to their initial values $x_1^{(0)}$, $x_2^{(0)}$, and $x_3^{(0)}$ (often values sampled from a prior distribution q). At iteration i, we sample $x_1^{(i)} \sim p(X_1 = x_1 | X_2 = x_2^{(i-1)}, X_3 = x_3^{(i-1)})$, sample $x_2 \sim p(X_2 = x_2 | X_1 = x_1^{(i)}, X_3 = x_3^{(i-1)})$, and sample $x_3 \sim p(X_3 = x_3 | X_1 = x_1^{(i)}, X_2 = x_2^{(i)})$.

The process continues until a possible "convergence" is reached (the sample values have the same distribution as if they were sampled from the true posterior joint distribution). Algorithm 1 details a generic Gibbs sampler [18].

Algorithm 1 Gibbs sampler

Initialize $x^{(0)} \sim q(x)$
for iteration $i = 1, 2, ...$ do
  $x_1^{(i)} \sim p(X_1 = x_1 | X_2 = x_2^{(i-1)}, X_3 = x_3^{(i-1)}), ..., X_D = x_D^{(i-1)}$
  $x_2^{(i)} \sim p(X_2 = x_2 | X_1 = x_1^{(i)}, X_3 = x_3^{(i-1)}), ..., X_D = x_D^{(i-1)}$

  $x_D^{(i)} \sim p(X_D = x_D | X_1 = x_1^{(i)}, X_2 = x_2^{(i)}), ..., X_D = x_{D-1}^{(i)}$
end for

Algorithm 1, simulates samples by sweeping through all the posterior conditionals, one random variable at a time (instead of directly sampling from the posterior distribution itself). In its early iterations, as random values are used to initialize the algorithm, the simulated samples might not necessarily represent the actual posterior distribution. However, the theory of MCMC guarantees that the stationary distribution of the samples generated under Algorithm 1 is the target joint posterior. This is why, MCMC algorithms are typically run for a large number of iterations to have a higher chance for achieving convergence [18].

2.2.3 Collaborative Filtering

Traditional Collaborative Filtering

Traditional collaborative filtering algorithm represents the user as an N-dimensional vector of items in which, the components of a vector are positive for positively rated items and negative for negatively rated items. To compensate for best-rated items and for making less well-known items more relevant, the algorithm is typically modified by weighting the vector components by the inverse frequency. The algorithm generates recommendations based on a few similar users and a common method for measuring this is; calculating the cosine of the angle between the two vectors:

$$similarity(\overrightarrow{A}, \overrightarrow{B}) = cos(\overrightarrow{A}, \overrightarrow{B}) = \frac{\overrightarrow{A} \bullet \overrightarrow{B}}{\|\overrightarrow{A}\| * \|\overrightarrow{B}\|}$$

Once a similar user is found, the algorithm selects recommendations from the found users' items by forming a rank for each item according to how many similar users rated it. This is computationally expensive and requires a lot of recourses (for very large data sets, the algorithm encounters severe performance and scaling issues). In the worst case it could be $O(MN)$, where M is the number of users and N is the number of items in inventory, since it examines M users and up to N items for each user. However, the algorithm's performance is actually closer to $O(M + N)$, because the average user vector is extremely sparse. Scanning through every user is approximately $O(M)$, since most user vectors contain a small number of rated items, regardless of the size of the inventory. But there are statistically a minority, whom have rated a significant percentage of items, requiring $O(N)$ processing time. The final performance is approximately $O(M + N)$ [20].

It is possible to partially address these scaling issues with the reduction of data-size M (by randomly sampling the users or discarding users with few ratings), and by reducing N (discarding very popular or unpopular items). It is also possible to reduce the number of items examined by a small, constant factor by partitioning the item space based on product category or subject classification [20].

Item to Item Filtering

"Rather than matching the user to similar users, item-to-item collaborative filtering matches each of the user's rated items to similar items, then combines those similar items into a recommendation list " [20].

The algorithm determines the best match by iterating through all the possible item pairs and computing a similarity metric for each combination, building a product-to-product matrix form the results. However, many product pairs have no common users, and thus the approach is inefficient in terms of processing time and memory usage. The following iterative algorithm fixes this by calculating the similarity between a single product and all related products:

Algorithm 2 Item Filtering

For each item in product inventory, I_1
  For each user C who rated I_1
    For each item I_2 rated by user C
      Record that a user rated I_1 and I_2
  For each item I_2
    Compute the similarity between I_1 and I_2

The similarity between two items is possible to compute in various ways, like with the previously mentioned cosine measure, switching each user vector component with a corresponding item, and the vector's M-dimensions correspond to users who have rated that item [20].

This method is exceedingly time intensive, with $O(N^2M)$ as worst case. In practice, however, its performance is closer to $O(NM)$, since most users have very few ratings (as previously explained). By choosing users who positively rated popular titles reduces runtime even further, but with reduction in quality. The algorithm finds items similar to each of the user's ratings, aggregates those items, and then recommends the most popular or correlated items. This computation is very quick, depending only on the number of items the user rated [20].

3 EXPERIMENTS

In order to evaluate the Trend-based Recommendation Model, one should compare it to a similar method using identical data set. The goal of this experiment is to show the difference between a regular recommendation method and Trend-based recommendation to pinpoint differences. This thesis will use an example to elucidate it futher, using candidates background information gathered form Avoindata.fi to create a recommendation model of election candidates for the voters.

Ideally, the best way to assure highly accurate recommendations would only be possible when using multiple parameters. However, this requires an access to sophisticated computers, without them it's very difficult to do all-encompassing calculations. The lack of calculative power, pushes the experiment to narrow down. This in mind, the experiment uses Helsinki Election Area with 4 to 7 parameters from 238 candidates instead of whole Finland to rank candidates for the average voter. The constricted set allowed more iterations within the time frame, and the set could now be easily manipulated to make the assumption that the average voter using Trend-based recommender is interested in the following factors:

-Which party is the candidate representing?
-Does the candidate have any children?
-What languages does the candidate speak?
-What is the candidates education level?

Using these factors Trend-based Recommendation Model will rank the possible candidates into an order which will then be compared to the results of a regular method.

The first phase of the experiment is carried out by creating the 3rd-order tensor from candidate data. This tensor will be used by both methods. The tensor will contain candidate's ID number (I), candidates answer to voters question (Q) and Election area (A) $X \in \mathbb{R}^{|I| \times |Q| \times |A|}$.

Next, Motif finding with Gibbs sampling is used to find looming trends within the tensor. In order to enhance the accuracy, the motif finding is executed to run a couple of times to get multiple chains. From these chains the repeatedly emerged is chosen for further enhancement.

In favor of relevant results, the chosen chain is scored and then ranked reducing the unnecessary candidates. The scores are distributed by factor-to-factor collaborative filtering and to the whole party is presented for the voter in priority order. This order will be compared to regular method which is just the order from collaborative filtering.

4 RESULTS

This section analyses the results of the experiments, the later section called Conclusion is dedicated to overall analysis and investigation.

The experiments went as expected with no unusual events that would have introduced error. Table 2 shows the measured ranking for factor-to-factor collaborative filtering and Table 3 for Trend-based Recommendation Model, the scores for each presented factor are shown in each cell of Tables. Convergence in the distribution was achieved within 10 iterations.

As part of this experiment, the voters choices were determined beforehand. Each ID number represents a single candidate. Factors in horizontal axis; Social Democratic Party (P), No Children (C), Other Languages than English, Swedish or Finnish (L) and University level Education (E), represent candidates answers to poll questions with a yes or no choice. In these calculations, the voters priorities are the following: $(E) > (L) > (C) > (P)$. These choices are depicted in both Tables as bolded characters. Other political parties are not represented in the results. The results shows a list or coordinate vector of recommendations for a specific candidate, the higher the score is the higher its priority is for the candidate.

Table 2. Top 10 election candidates by Collaborative Filtering

ID number	**(P)**	**(C)**	**(L)**	**(E)**
5777	117	35	65	82
5627	117	35	65	82
4883	117	35	65	82
4772	117	35	65	82
4400	117	35	65	82
5455	111	43	63	80
4812	111	43	63	80
4486	111	43	63	80
4315	111	43	63	80
4411	107	31	55	73
...	...	...	...	...

Table 3. Top 10 election candidates by Trend-based Recommendation Model

ID number	**(P)**	**(C)**	**(L)**	**(E)**
5868	0	5	10	11
4280	0	3	20	10
4411	0	3	20	10
4413	0	8	0	10
4488	0	8	0	10
5560	0	7	10	7
4387	0	3	10	6
5711	0	3	10	6
4315	0	1	30	5
5777	0	1	30	5
...	...	...	...	...

5 CONCLUSION

The objective of this thesis was to combine various methods to build a Trend-based recommendation model and study the results by comparing. Overall, it is difficult to argue which method is better without end-user/voter involvement, though a comparison in factual basis between the Tables 2 and 3 reveals clear differences in Trend-based models favour:

Firstly, Trend-based recommendations could be considered to be more accurate since it focuses its scoring in three factors it considers to be motifs instead of four, though without a real-world user testing it's hard to judge which of the results satisfies the voter most. By looking at Table 3's scores, it can be seen that Trend-based recommendation model didn't consider the political party (P) factor important enough to be part of the overall motif and therefore scoring. On the other hand Table 2's highest scores are in (P) factor, even though the experiment was limited to the same political party and was meant to be the least important factor. This can be explained as the motif finding with Gibbs sampling tries to find common variations. Since all the candidates have the identical (P) factor it's not considered a variation nor an underlying trend.

Secondly, Collaborative filtering gave the Top 5 candidates identical scores (depicted in Table 2), which re-confirms that in general the results are difficult to score naturally and are usually without any serendipities. The problem of ranking identical scores happens especially when using simple scoring methods such as collaborative filtering. Table 3 on the other hand appears to have a more variety in scores, having only few common candidates with Table 2. This is clearly due to the focus on found motifs via gibbs sampling and it shows by a semi-natural scoring.

Taking these mentioned differences into account, it could be concluded that the Trend-based recommendation model surpasses the regular model in score diversity, natural parsing and plausibly in accuracy. However it is not without its faults and the model's complexity might be a big disadvantage compared to other regular models such as collaborative filtering. Trend-based recommenders complexity is higher than collaborative filterings and the bigger the data-set, the longer time gibbs sampling requires to finish its iterations.

Nevertheless, by focusing in hidden patterns within the data Trend-based recommendation model shows promise, and rather than making the model less complex it could be improved by choosing the factors some other mean than through the voter's interest. An alternative could be to use the model to pinpoint a starting point for a faster recommender system, using Trend-based model as a locating model instead or just simplifying the data-set to create multiple targeted recommendations.

No matter who or which variation of the model is going to be used or used by, the model is showing a view of the data-sets underlying trends. By purely speculating, this kind of a model could potentially be a great benefit for a variety of industries and fields. From analyzing Big Data [21] to speeding up machine learning process [22] or perhaps finally, realizing the personalized ecosystem of information to which the trailblazer are steadily progressing towards to.

References

[1] B. Darrow, *Amazon and google continue cloud arms race with new data centers*, Available from: http://fortune.com/2016/09/30/amazon-google-add-data-centers/, [Accessed 2-October-2016], 2016.

[2] A. Glaser, *Google, facebook, amazon and others have teamed up to make ai seem less creepy*, Available from: http://www.recode.net/2016/9/28/13098780/google-facebook-amazon-ibm-nonprofit-ai-creepy-artificial-intelligence, [Accessed 2-October-2016], 2016.

[3] D. Markovikj, S. Gievska, M. Kosinski, and D. Stillwell, "Mining facebook data for predictive personality modeling," in *Proceedings of the 7th international AAAI conference on Weblogs and Social Media (ICWSM 2013), Boston, MA, USA*, 2013.

[4] H. Choi and H. Varian, "Predicting the present with google trends," *Economic Record*, vol. 88, no. s1, pp. 2–9, 2012.

[5] B. Pan, H. Hembrooke, T. Joachims, L. Lorigo, G. Gay, and L. Granka, "In google we trust: Users' decisions on rank, position, and relevance," *Journal of Computer-Mediated Communication*, vol. 12, no. 3, pp. 801–823, 2007.

[6] CBInsights, *The Race for the AI: Most Active Acquirers In Artificial Intelligence*, Available from: https://www.cbinsights.com/blog/top-acquirers-ai-startups-ma-timeline/, [Accessed 29-June-2016], 2016.

[7] J. C. Feng, "The poor man's introduction to tensors," *Wolfram Mathematica Student Edition*, p. 17, 2015.

[8] Wikipedia, *Collaborative filtering*, Available from: https://en.wikipedia.org/wiki/Collaborative_filtering, [Online; accessed 2-October-2016], 2016.

[9] Yleisradio, *Eduskuntavaalien 2015 ylen vaalikoneen vastaukset ja ehdokkaiden taustatiedot*, Available from: https://www.avoindata.fi/data/fi/dataset/eduskuntavaalien-2015-ylen-vaalikoneen-vastaukset-ja-ehdokkaiden-taustatiedot, [Online; accessed 2-October-2016], 2015.

[10] Y. Sakurai, Y. Matsubara, and C. Faloutsos, "Mining and forecasting of big time-series data," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ACM, 2015, pp. 919–922.

[11] S. M. Lynch, "Modern model estimation part 1: Gibbs sampling," in *Introduction to Applied Bayesian Statistics and Estimation for Social Scientists*, Springer, 2007, pp. 77–105.

[12] J. Hu, B. Li, and D. Kihara, "Limitations and potentials of current motif discovery algorithms," *Nucleic acids research*, vol. 33, no. 15, pp. 4899–4913, 2005.

[13] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, J. Riedl, *et al.*, "Combining collaborative filtering with personal agents for better recommendations," in *AAAI/IAAI*, 1999, pp. 439–446.

[14] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[15] D. W. McDonald and M. S. Ackerman, "Expertise recommender: A flexible recommendation system and architecture," in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, ACM, 2000, pp. 231–240.

[16] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[17] X. Tang, Y. Xu, A. Abdel-Hafez, and G. Shlomo, "A multidimensional collaborative filtering fusion approach with dimensionality reduction," 2014.

[18] I. Yildirim, "Bayesian inference: Gibbs sampling," *Technical Note, University of Rochester*, 2012.

[19] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to mcmc for machine learning," *Machine learning*, vol. 50, no. 1-2, pp. 5–43, 2003.

[20] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[21] H. Chen, R. H. Chiang, and V. C. Storey, "Business intelligence and analytics: From big data to big impact.," *MIS quarterly*, vol. 36, no. 4, pp. 1165–1188, 2012.

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