University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Łukasz Kidziński
Student id 234151

Statistical foundations of recommender systems

Master's thesis
in MATHEMATICS
in the field of APPLIED MATHEMATICS

Supervisor:
Ph.D. Hung Son Nguyen
Institute of Computer Science

September 2011

Supervisor's statement

Hereby I confirm that the present thesis was prepared under my supervision and that it fulfils the requirements for the degree of Master of Computer Science.

Date Supervisor's signature

Author's statement

Hereby I declare that the present thesis was prepared by me and none of its contents was obtained by means that are against the law.

The thesis has never before been a subject of any procedure of obtaining an academic degree.

Moreover, I declare that the present version of the thesis is identical to the attached electronic version.

Date Author's signature

Abstract

The recent extensive growth of information available on the Internet, additionally speeded up by a social networks, have evoked the need for systems which can help users to find valuable information. In this context Information Filtering provides tools which support users in their searches. In this thesis I developed strong mathematical background for recommendations and proposed a specific system built on the developed up foundations. The work is divided into 4 main parts and constitutes a survey over whole methodology of building effective engine: definition of the problem, selection of the evaluation measure, choice of proper algorithms and validation. Two experiments were conducted for illustrating and testing various methods. First concerned gathering and predicting ratings of sweets with use of collaborative-filtering techiniques. Second, the real-life experiment, was conducted on venues' search website Neib.org for increasing the users' avarage time spent on the site.

Keywords

recommender system, collaborative-filtering, matrix completion, singular value decomposition, principal component analysis, model evaluation, reinforcement learning, data mining

Thesis domain (Socrates-Erasmus subject area codes)

11.2 Statistics
11.4 Artificial intelligence

Subject classifcation

62. Statistics
62.P. Applications
62.P.99. General

Title in Polish

Wykorzystanie metod statystycznych w systemach rekomendacji

Contents

Introduction

"If I have 2 million customers on the Web,
I should have 2 million stores on the Web."
Jeff Bezos, CEO of Amazon.com

The recent extensive growth of information available on the Internet, additionally speeded up by a social networks, have evoked the need for systems which can help users to find valuable information. In this context Information Filtering provides tools which support users in their searches. The goal of these systems is to rank data according to partly revealed preferences, which are encapsulated in previously undertaken surveys, marks users' provided and in information properties itself. Most efficient exploitation of that data is an object of many recent studies, where the research is accelerated by ecommerce corporations.

In this thesis I developed strong mathematical background for recommendations and I proposed a specific system built on the worked up foundations. The work is divided into 4 main parts. Chapter 1 contains a formal introduction to various aspects of the problem, including user or item features as well as motivations and problems related with those techniques. The following part (chapter 2) contains a specific approaches for the evaluation of a recommeder models. This is where I mathematically expressed goals of well-designed system. Then in chapters 3 and 4 I provided mathematical tools for building recommender systems and data mining methods which further exploit those techniques.

In the final part of my thesis two cases are presented for illustration of the topic. First one is the sweets recommender system, which I have written for resembling the Netflix prize, the worldwide contest held in 2009. Since the prize constituted $1mln reward it attracted a plethora of brilliant minds to contribute, in consequence dozens of new algorithms were discovered. The second case study presents neib.org rocommender system for local venues which constitutes the ultimate goal of this thesis.

Neib.org website was set up in 2009 by two students of University of Warsaw, Łukasz Kidziński and Paweł Bedyński. It is the social platform where users review and rank local venues. Till July 2011 over 11000 reviews were added and every day over 2000 unique users visit site and look up for suggestions. In general, effort of the recommender system in this domain aim at maximization of users satisfaction and extension of their time spent on site which in turn leads to increased users loyalty.

Chapter 1

Problem definition

It was just in mid-1990s when a recommender systems became an independent research area [19]. Since then interest has explosively increased due to applications in ecommerce. Extensively active research are illustrated by following facts:

1. In an abundance of huge web portals like Amazon.com, Yahoo, YouTube or Netflix, recommendations play an important role in retaining users present on site or just directly increase company's profit. In particular other big websites like last.fm or jinni.com, recommender constitutes a key feature itself.

2. There are a conferences and workshops devoted to this field. The most popular one is the ACM Recommender Systems (RecSys). It was first held in 2007 and since then it became main annual event in recommender technology and applications.

3. All over the world courses dedicated entirely to RSs are provided on undergraduate or graduate level. There are many blog posts and tutorials for beginners to provide recommenders on their websites. Additionally the books about RS techniques have been published recently [1].

4. The special issues of academic journals devoted to RSs were published, the examples are as follow AI Communications (2008), IEEE intelligent Systems (2007), ACM Transactions on Computer-Human Interaction (2005) and many others.

1.1. Goals

One of the reasons why RSs became that popular is the fact that effects are relatively easy measurable, and results after injecting even simple model are usually significant.

Motivations for using RSs include:

- *Increase volume of sales.* Most popular use of RSs since it gives instant, and in some cases very significant, profit. Netflix, the on-line video rental, set up a contest for improving it's system by 10% with reward of 1 million USD what consists a clear prove of how valuable this research is for huge companies.

  Non-commercial institutions have usually the same goal, however, in this case money is not directly associated. For instance Google suggests best articles in it's search for getting user's loyalty and trust.

- *Diversity of sales.* This one also involves Netflix, however, it is not related to organized contest. Basically, company aims at renting all DVDs from the catalog, which accounts for not only the most popular once. On the other hand it does not aim at promotion of the movies which users are not likely to watch, so well-designed RS is a superb tool for this task.

  Another case is banning "too popular" content, what can be observed in Yahoo!. In circumstances when unwelcomed articles become popular the website performs a special activities to decrease the number of viewers to avoid the identification of the website with a tabloid.

- *User's satisfaction.* The higher user's satisfaction is, the more loyal he becomes. This comprises the reason why we may want the user to be implicated in RS for as long as it is possible. Secondly, if the user interacts more, system gets more information about him, which in consequence may be turned into new data. Furthermore, this leads to fulfilling other goals even better.

- *Gathering user's data.* Collected data may be used not only for future recommendations but also for other aims, for instance tuning up the product or setting up special promotions.

  Main motivation for working on RSs is a social assumption that people rely on suggestions from others [8]. Exploiting of users' preferences may help to select products which they are more likely to buy, hence to increase companies profit.

Carol Bartz, the CEO of Yahoo!, after installation of recommender system on main site, said: "The result was a 100% click-through rate increase over time. In January [2009], we introduced release 2 of the engine, which included some of our behavioral targeting technology. This capability - coupled with great content - led our Today Module to experience click-through rates 160% over pre-engine implementation." [9]

1.2. Variants of applications

There are dozens of variants of recommenders that one can apply [1].

- *Find some good items.* This is an usual task in e-commerce systems, when one aims to show user a particular items that he may be interested in.

- *Find all good items.* In some cases it is not the descending order of items, what we are looking for, but just set of them which are above some threshold value. User may intend to look for instance for restaurants which present defined level of quality. After picking the set, user searches for details of them on his own.

- *Annotation in context.* In this case the goal is to mark items in context as interesting. An example of this may be TV programme where we want to mark shows or movies valuable for given user.

- *Recommend a sequence.* In this circumstances one looks not only for a particular item but for whole set of them. There are users that may be interested in books related in some extent with each other, like collection of anti-utopian issues.

- *Recommend a bundle.* This kind of recommendation helps user to configure a whole set. When one aims to build his own PC-set, recommender tells which graphic card,

CPU, hard drive and main-board use together. It differs from choosing just a sequence, since items posses far different properties and it is the combination which gives the real value.

- *Just browsing.* Another task may concern the retention of user presence on site. In this case the effort was to indicate diverse items just to keep user focused, since this increases probability that eventually he will find something interesting.
- *Tune up recommendation.* Users may not believe in recommendations or want to get them more diverse, hence one of requirements might concern equipping the system with some parameters adjustable manually.
- *Profile information.* Usually what is desirable is to persuade user to present his personal preferences either in a survey or just by inputing a query string.
- *Express self.* There are users interested in using recommender as a way to express their own views and believes - especially when they are enabled not only to provide marks but also to write reviews. They are satisfied just by posting something and knowing that other users will read it.
- *Help others.* There are people who strongly aim to help others and are bona fide satisfied whenever they can contribute to community. If they are not happy with car already bought, they look for occasion to express it in community interested in acquiring that car.
- *Influence others.* There are known cases where people want just to bias recommendation to influence others to acquire particular products or services. Some companies want to promote themselves or revile others.

1.3. Formal definition

Let me start with some basic technical assumption. First of all I want two instances of the same item to be treated equally, the example could constitute two CD copies of the same album. That is why I define Item as following

Definition 1. *Item is class of objects of the same kind, indistinguishable by the user, with at least one instance.*

Secondly, I assume rather e-commerce applications. We are not going to cover some other decision aids aspects for companies or institutions. Let us define

Definition 2. *User is person enabled to provide his own preferences by specifying them exactly in some survey or by evaluating items.*

I will represent utility function for each user as vector of values over items. Set of those vectors form matrix.

Definition 3. *Preferences matrix M is $n \times m$ matrix where n is number of items and m is number of users with integer entries from set* $[1, 5]$.

Definition 4. *Property is a feature which has a numerical or categorical representation and can have impact on user's decision. With each property p we define set of possible values as V_p.*

Definition 5. *Annotation function is function for property p,*

$$a_p : S \to V_p$$

where S is set of items.

Since set P of properties is finite we may identify annotation for each item with vector of length $|P|$.

In this model one can describe presence of WiFi in mobile phone as property with categorical values *yes* (1) or *no* (0). For other features, especially those for which expert's information is needed, I use fuzzy logic ideas, such that for instance movie *Sweeney Todd* is 0.7 horror movie.

What is usually given is partially revealed preference matrix $\hat{M}$. The ultimate goal is to present to the user K items he is most likely to buy. First thing is to reveal preferences and then just to sort items in descending order.

Additionally there can be information available in user's input, for instance his mood, which is the case regarding jinni.com website. There are also recommenders which do not take user's data at all - for instance TOP10 (of LCD, washing machines, cars, etc.) in magazines. However this kind of suggestion is usually based on expert's knowledge and is not considered as the area of RS research [1, p. 11].

The last thing needed is definition of input or context in which one is looking for recommendations. This may be for instance user's query string, which is the case in Google, or some item's page for which one aim to show similar, and preferred by the user, links. Idea of modeling this is similar to item's annotations.

Definition 6. *User's context is a vector of properties.*

1.4. Problems to overcome

During process of building system one should be aware of some difficulties which may appear.

First problem which one need to deal with is defining the way of system evaluation. There exist two aspects of it: practical and theoretical one. As practical one means taking into account feature that one is mostly interested in. On the one hand one may want to be as close as possible to user's real preferences, but on the other it may be more important just to maximize the profit, and those two goals not always go together. As a theoretical aspect I mean mathematical definition of measure which should be consist and easy to manipulate, like square distance instead of absolute value.

In content-based problems one need to deal with retrieval of properties of an object. This task is easy for items like mobile phone or computer where we have just a metric, but it becomes much harder whenever it comes to annotate music or movies and becomes almost impossible with some abstract pictures for instance.

In some cases it is crucial to provide consistent expert data in order to get valuable recommends. Good example for this is pandora.com, where experts have annotated over 800 000 tracks, taking into account gender, instruments, artist, dynamics and many others, not revealed officially by Pandora. Lots of expert's work was needed not only in providing data but also in designing the parameters.

Another problems are related to utility theory. It appears that rational assumption is sometimes broken. For instance there are cases where transitivity of preference is unsatisfied. One may treat movies A and B as of the same quality, but prefer C to B without preferring C to A. It creates the problem especially in systems based on one-to-one comparisons.

While applying machine learning methods usually we need to deal with common statistical problems which may include:

- lack of knowledge about additional unknown explanatory variables,
- lots of possible variables - we can store each user action but a priori we do not know if it is useful (a day of the week of rating the item, user's time on site, time between using and rating, etc.),
- since we usually use mean square error as quality measure, it may happen that we pay to much attention to outliers,
- users interprete scale 1-5 in different ways - there are people who put mark 5 very generously while others put it only for few best movies,
- user's rank may be biased by some unmeasurable (hence random for the system) conditions like mood during watching,
- problems with dealing with large datasets, sparse matrices, matrix multiplication and computation complexity in general.

And as other statistical models, recommender system can be biased by users who ranks more often than others, by popular items with lots of ranks or just by cheaters.

During research on Netflix prize, it appeared that there are items for which it is hard to predict anything. One of them is Napoleon Dynamite - people love it or hate it, although, it is hard to predict their preferences basing just on other votes. Problem with hard prediction additionally appears with new items which do not have any marks at all.

Last thing to mention is that domain really counts - risk of prediction varies very much on it. There is no problem if our RS suggests not the best music track, but the risk is much higher whenever it comes to house or car recommendation where dissatisfaction can be much more profound.

1.5. Related problems and fields

Due to interdisciplinary nature of the recommender system problem, research is based on variety of approaches. Recent studies show some applications in decision-aids like detection of leukemia [21] or security alert dissemination [20]. Calling it "recommender system" is probably little misleading, however, research in both areas is strongly correlated.

Game theory and voting ideas are often used to prove some theoretical results about recommendation systems [20][22].

Commercial use of RSs in search engines also include context adverts like in Google Search. Recent studies have additionally included building economic models [23].

Research in other fields of statistics and machine learning may appear to be contribution to recommender systems and vice-versa. One may think of RSs as some other point of view on well-known information filtering problems. In following two chapters I will focus on statistical and machine learning techniques which appeared useful in the process of revealing user's preferences.

Chapter 2

Evaluation of Recommendation System

Before proceeding to the algorithm building process, a method of evaluating is needed. First in the section 2.1 I present three approaches to the process of evaluation: off-line, user studies and on-line experiments, according to the Microsoft's paper [26]. Then I focus on the various properties that one wants to optimize. In the last section the new approach based on the utility function is presented.

2.1. Experimental Methodology

Methodology for conducting the experiments is common for all empirical studies like physics, psychology, sociology, economics, etc. This is the reason why the wide selection of literature is available [28]. In this section I briefly describe three main ideas used in the evaluation of a recommender system.

In every one of this three cases, we need to follow well-known general principals

- On the beginning of the experiment hypothesis should be fixed and experiment should be conducted only for accepting or refusing that hypothesis,
- which are not tested should also be fixed or randomized.
- One need to be sure that results can be easily generalized for population.

There are dozens of other "rules of thumb" and processes need to be followed in the experiment. These processes are well defined in the literature, e.g. in [31]. Here I specify three frameworks used for evaluation recommender systems.

2.1.1. Off-line experiment

Since off-line experiment is the only one which do not require any user's interaction, it is the easiest one to conduct. In this case the dataset of users, items, rates and properties is given. In a standard scenario we

1. select one user,
2. hide some of his ratings,
3. try to predict them,

4. compare predictions with the real values.

This methodology is similar to k-fold cross-validation or leave-one-out method [29].

2.1.2. User studies

The method is based on survey of randomly selected users. There have been made a wide research, in the field of tests preparation [30], mainly for social studies, which can be applied here.

In this case cost of running the experiment is much higher, therefore usually before the test starts one try to drop the worst candidates by much cheaper off-line method or detailed expert analysis.

One important thing is that the group, on which test is conducted, need to represent accurately the target group. Moreover one should try to gather as much data as possible, to avoid the need of survey repetition for testing new variables.

In user studies, frequently, the algorithms are assigned at random, individually to each user. Then the performance of the algorithm among the group of users, as well as the differences between algorithms can be compared. Described methodology is often referred as **A-B testing (All-Between)**.

2.1.3. On-line experiment

In bulk of applications it is the interaction between the user and the system, which is highly important. In this circumstances data need to be collected on-line, i.e. one run recommender system for the user who is not aware of being tested.

The main drawback of this setup is that very bad algorithm may annoy the user, so that he would not use it again in future. That is the reason why one usually use off-line experiment first, to choose the best algorithms, and subserviently test only the preselected ones on-line.

One of the advantages is that this way almost every feature is testable. In particular it offers the opportunity of estimation of profits from usage of different algorithms directly.

2.1.4. Two-step approach

In general it is assumed, that items which should be shown as recommended, constitute the ones which the user is most likely to use. However, it might be important to split the action into two steps:

1. clicking the recommended item,

2. using (acquiring) it.

Taking this under consideration, the recommendation process can be understand as two subsystems, one which selects items the user is most likely to click on, and the second one which select items he is most likely to acquire. Finally, finding the intersection of those two sets comprises the ultimate goal. This idea leads to the new approach to the evaluation process and algorithm building. However, since only the on-line evaluation may be used, it is relatively hard to apply. What is needed is the information if the given item was bought directly after recommendation by the system.

2.1.5. Drawing conclusions

The main goal of the evaluation focuses on choosing the best algorithm. Algorithms can be ranked according to selected measure, however, in most cases it is sufficient to compare algorithms in pairs. For this purpose standard statistical methodology and significance tests can be applied.

The common tools for measuring significance are McNemar's test, paired t-Student or Wilcoxon signed rank test. The procedure is similar in each case: First, one assumes that there is no significant difference and that the deviations are random, then if it appears that the event that occurred had very small probability (below some threshold $\alpha = 0.05$ or smaller) we reject the hypothesis that there is no difference. The theory of statistical tests has been developed and well documented [28].

The simplest (and surprisingly very efficient) example is sign test, where one try to prove that algorithm A is better than B. In order to do so, one assumes that in each particular prediction difference (i.e. which algorithm is better) is random with probability $p = 0.5$. Foreasmuch, if one have n predictions, the outcome is expected to follow Bernoulli distribution, i.e.

$$\mathbb{P}\left(success > n_A | A = B\right) = 0.5^n \sum_{k=n_A}^{n} \binom{n}{k} \tag{2.1}$$

where n_A is the number of cases where A outperforms B. Whenever this value is lower then threshold α we say that A is significantly better than B.

Nevertheless, comparisons in pairs is not always adequate. In this setup probability of wrong decision is at level 5%. If we compare 10 pairs then the chance of mistake is equal to $1 - (1 - 0.05)^{10} > 0.4$. To overcome this problem we use **ANOVA** or **Friedman's test for rating** which have also been studied in great detail [32].

2.2. Properties of Recommender System

Initially recommender systems have been evaluated according to their ability to predict user's preferences - the more accurate prediction had been the better algorithm performed. Notwithstanding, it became clear that good prediction is not always everything that users expect. People are additionally interested in exploring unknown items, preserving their privacy, minimizing the risk of dissatisfaction, getting fast response and bulk of other features. Secondly, it befalls that for the ecommerce company, not the user's satisfaction but the utility (like customer's health for pharmacy or profit for e-commerce organization) is the most important value.

In their recent work Guy Shani and Asela Gunawardana [26] have summarized properties to optimize. Some of them are presented in following sections

2.2.1. Prediction Accuracy

The most widely discussed property in the recommendation systems is the accuracy of prediction. The engine may predict two features: user's opinion (e.g. the rating) or the probability that user will take an action (e.g. acquisition). Accurate prediction is frequently the goal of the research, it offers the better understanding of the user's preferences and helps in optimizing other features of the RS.

One can distinguish three classes of accuracy measures, which have roots in statistics. Accuracy can be measure in context of

- ratings predictions,
- usage predictions,
- rankings of items predictions.

Ratings accuracy

The most common way of measuring the prediction error is **Root Mean Squared Error (RMSE)**. For user u and item i in the test set $(u,i) \in \mathcal{T}$ we derive the prediction $\hat{r}_{u,i}$ and compare them with known values of $r_{u,i}$ with the following formula

$$RMSE = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} \left(\hat{r}_{(u,i)} - r_{(u,i)}\right)^2}.$$

Since, the square function is monotonic, it is common to omit the root and use the $MSE = RMSE^2$ instead.

Common alternative for the RMSE is the **Mean Absolute Error**

$$MAE = \frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} \left|\hat{r}_{(u,i)} - r_{(u,i)}\right|.$$

Main difference between those two values is that $RMSE$ punishes harder large errors. For instance the RMSE method prefers system which makes an error of 2 on two items to the one that makes error of 3 on one item and 0 on the second, while MAE would prefer the opposite.

Sometimes it is convenient to use a normalized version (i.e. by the range of scale $r_{max} - r_{min}$), since, it might be easier to compare algorithms out of context and it does not change the order.

Another variant, called the **Average RMSE**, makes use of a weighted sum, i.e.

$$ARMSE = \sqrt{\sum_{(u,i)\in\mathcal{T}} w_i \left(\hat{r}_{(u,i)} - r_{(u,i)}\right)^2},$$

where $w_i > 0$ is the importance of item i (which may be it's popularity for instance) and $\sum w_i = 1$. It is used frequently in balanced datasets, where particular items appear very rarely.

Instead of using just a difference $\left|\hat{r}_{(u,i)} - r_{(u,i)}\right|$ for punishing bad prediction, other distances can be introduced (in particular asymmetric functions for distinguishing statistical errors of Type I or II).

Usage accuracy

In vast number of applications, the goal is defined not as prediction of user's ratings, but as probability of taking an action. For example, in case of DVDs distribution, it is more important to impel the user to buy an object, then to find items that he likes the most.

In evaluation of such system, one typically chooses some test user, hide several random selections and try to predict the set of items that the user would like. Table 2.1 presents possible outcomes of such recommendation.

It is important to mention that there is an assumption that if the user did not pick the particular item he is not interested in it. Nonetheless, this assumption may be false - it may

	Recommended	Not recommended
Used	True-Positive (tp)	False-Negative (fn)
Not Used	False-Positive (fp)	True-Negative (tn)

Table 2.1: Classification of possible outcomes of a recommender system

happen that the set of suggestions contains something interesting what user have not seen before. This is the reason why False-Positive value is usually overestimated.

In order to compare algorithms one can define following statistics

$$\textbf{Precision} = \frac{\#tp}{\#tp + \#fp} \tag{2.2}$$

$$\textbf{Recall (True Positive Rate)} = \frac{\#tp}{\#tp + \#fn} \tag{2.3}$$

$$\textbf{False Positive Rate} = \frac{\#fp}{\#fp + \#fn} \tag{2.4}$$

$$\textbf{Specificity} = \frac{\#fn}{\#fp + \#fn} \tag{2.5}$$

Trade-offs between those quantities are frequently observed. For instance, longer recommendation list increases recall, yet, it also reduces the precision.

For the fixed length of recommendation list the precision of algorithms can be compared. Nonetheless, the quality of prediction vary on the size of list and choosing the best number is not an easy task. Thus, one usually calculate and plot the trade-off between precision and recall or between true positive rate and false positive rate. The former curve is called the **precision-recall curve**, while the latter one is known as the **Receiver Operating Characteristic (ROC curve)**.

Use of the ROC curve or the precision-recall curve is determined by the domain. In applications where cost of false positive is relatively high, one tries to minimize false positive rate and that is where ROC curves are used. This may concern for instance the company which sends free gifts for attracting new customers. On the other hand, in applications where cost of false prediction is relatively small (for example DVD recommender in video rental service) increasing quality of prediction might be more important than minimization of false predictions and that is where one uses ROC curves.

After choice of the method one computes curves for two algorithms. If one of them is strictly determined by the other then the choice is simple. However, when they intersect, the decision depends on the application. All the same algorithms can be still compared independently of the application by using measures like the **F-measure** [31] and the **Area Under the ROC Curve (AUC)** [33], but it is preferable to make selection based on the properties of the specific task.

Ranking accuracy

In most applications one wants to present a fixed list of recommendations. Two previously described methods achieve this goal indirectly and the final outcome comprise the ranking. In this section I present methods of evaluating returned ranking itself.

One of the typical methods is comparing predicted ranking with an reference one. If the ratings are given, as the reference ranking one can take the list of movies ranked by the user and sorted in decreasing order. Since we do not want to penalize resulting prediction

for ordering items which tie in reference rating, we use a measure like the **Normalized Distanced-based Performance Measure (NDPM)**, specified below.

Let me define r_{ui} as the reference rankings, $\hat{r}_{ui}$ as the system ratings of n_u items i for user i and

$$C^+ = \sum_{i<j} \operatorname{sgn}(r_{ui} - r_{uj}) \operatorname{sgn}(\hat{r}_{ui} - \hat{r}_{uj}) \tag{2.6}$$

$$C^- = \sum_{i<j} \operatorname{sgn}^2(r_{ui} - r_{uj}) \operatorname{sgn}(\hat{r}_{uj} - \hat{r}_{ui}) \tag{2.7}$$

$$C^u = \sum_{i<j} \operatorname{sgn}^2(r_{ui} - r_{uj}) \tag{2.8}$$

$$C^s = \sum_{i<j} \operatorname{sgn}(\hat{r}_{ui} - \hat{r}_{uj}) \tag{2.9}$$

$$C^0 = C^u - (C^+ + C^-). \tag{2.10}$$

C^u is the number of pairs for which the strict domination exists in reference ranking, C^+ and C^- are the numbers of pairs in correct order and incorrect order respectively, while C^0 is the number of pairs for which system ranking does not tie, while the reference one does. Finally, let me define

$$\mathbf{NDPM} = \frac{C^+ + 0.5C^0}{C^u} \tag{2.11}$$

Ties in reference ranking appear naturally in systems where user rank items by giving an integer mark. However, sometimes the preference ranking is exactly specified (for instance by providing binary choices between items). Then, one can use other measures like the **Kendall's** τ

$$\tau = \frac{C^+ - C^-}{\sqrt{C^u}\sqrt{C^s}} \tag{2.12}$$

or the **Spearman's** ρ

$$\rho = \frac{1}{n_u} \frac{\sum_i (r_{iu} - \bar{r})(\hat{r}_{iu} - \bar{\hat{r}})}{\sigma(r)\sigma(\hat{r})} \tag{2.13}$$

where $\bar{\cdot}$ and $\sigma(\cdot)$ denote the mean and the standard deviation.

2.2.2. Coverage

Next important feature of recommender system is the coverage. It might be that algorithm predict very wisely popular items, but it does not ever show rare or controversial ones. The same may be the case with algorithm which runs properly for users who have ranked many items and very poor for those who have not.

Popular quantity for measuring inequality in proposals is the **Gini Index** defined by

$$G = \frac{1}{n-1} \sum_{j=1}^{n} (2j - n - 1) p(i_j), \tag{2.14}$$

where $p(i)$ is the probability of appearing as recommendation and items i_j are ordered according to the increasing $p(i)$. The **Shannon Entropy**, given by

$$H = -\sum_{i=1}^{n} p(i) \log p(i) \tag{2.15}$$

is also used as the measure of disproportion. H is equal to 0 when the single item is always chosen and $\log n$ if probabilities are distributed equally.

In similar way one can compute inequality in click rate among shown recommendations by substituting $p(i)$ with the frequency of clicks.

2.2.3. Other measures

There are bulk of other measures which should be taken under consideration while tackling specific problem. Several of them are summarized in the table 2.2. Details can be found in [1].

2.3. Utility function and profit maximizing

Recent studies focus not only on the ability of prediction but also on the revenue of the whole system. For instance, from vendor's point of view it might be more important to make the profit on more expensive products, than to get good predictions on average. Das, Mathieu and Ricketts proposed [27] the model with the additive utility function, methods of evaluation and procedures for maximizing the profit.

The model assumes that the user trusts in a prediction with high probability if and only if he notices similarity of proposals with his own preferences over the fixed threshold value. Then, the goal is to find predictions which are valuable for the user but maximize the profit. For the user rank vector r, predictions $\hat{r}$, and the trust function $T(r, \hat{r}) \in [0, 1]$ one wants to maximize over $\hat{r}$ following

$$c \cdot \hat{\phi}(\hat{r}), \text{ such that } \mathrm{T}(\hat{\mathrm{r}}) > \alpha, \tag{2.16}$$

where α is the threshold trust, c is the profit vector and $\phi(\cdot)$ is the prediction of acquire.

The first thing to notice is that in this model, system does not try to give the best possible predictions, what may appear unethical. On the other hand, the condition $T(\hat{r}) > \alpha$ assures that proposals are still relatively good.

As the trust measure authors propose the **Dice measure**

$$\mathbf{Dice}(\hat{r}, r) = \cos\theta \frac{2\|r\|\|\hat{r}\|}{\|r\|^2 + \|\hat{r}\|^2} \tag{2.17}$$

where θ is the angle between vectors r and $\hat{r}$.

Now, for maximizing the profit over set $Dice(\hat{r}) > \alpha$ one can use the **Lagrange multipliers** [37]. Let me present it on simple example assuming that probability of buying the item i does not depend on buying other things and that

$$\hat{\phi}(\hat{r}) = \left(\frac{\hat{r}_1}{m}, ..., \frac{\hat{r}_n}{m}\right), \tag{2.18}$$

Confidence	Confidence describes the system's trust in its own recommendation. User can benefit from such information, for instance when the system recommends two items with the same value but the different confidence interval, user can add one item directly to the cart and read more specific information about the latter one.
Trust	This feature refers to user's trust in the recommendation. For instance showing him items which he already likes may be beneficial. Although such recommendation gives no additional value, it may increase his trust and interest.
Novelty	This property concerns system's ability to adopt to the change of preferences. For instance user who used to like Science Fiction movies could have changed his interest to comedies. The way to measure this quantity is to split the set of rates into two subsets by fixed date, hide all ratings after that date and some of ratings before. Then, apply the learning and evaluate with penalizing the system for making recommendations of items liked before the splitting date, i.e. in the past.
Serendipity	Serendipity tells how surprising the successful recommendation is. It refers to the amount of additional information relevant for the user. In order to get better results in this factor, one can reward systems which give recommendations laying "far" from already ranked in the item space. As the distance measure the number of different features (such as author, genre, etc.) can be used.
Robustness	Robustness is the system's awareness of the fake information. As studies show [34], more people rely on the information from Internet. That is why PR and marketing companies try to bias recommendation by promoting their products. While it is impossible to build the system completely immune to attacks, one can try to use some methods to avoid using distrustful information (such us many ratings provided from the same IP or dozens of similar users who ranked only few items in the same time). Unfortunately, this feature is hard to measure and evaluate, since one can defend the system only against particular attacks.
Privacy	It is inappropriate to reveal any private information for a single user. In order to avoid disclosure, any theoretical scenario in which user's preferences leak to the public (or to the user with similar preferences) should be checked and eventually blocked.
Speed & Scalability	In vast majority of applications, the ability to scale to the large dataset is the key feature. For systems which provide recommendations on-line the speed is crucial. Ability to parallel computations is also considered factor in applications.
Ease of interaction	Since recommender systems are designed for people, they are expected to be easy to use. It may happen that asking a few questions gives information that provide great abilities but yet, it may annoy the user. This property is rather subjective, therefore, it is measured by user's survey.
Recommender's utility	The common way to summarize trade-off between all properties or emphasize on one feature is making use of utility function. One example of additive utility function is presented in the section 2.3.
User's utility & risk	We can also try to estimate the user's utility although, it may be hard to model. If one manages to grasp the performance function, he can also calculate the risk and penalize (or award) hazardous proposals.

Table 2.2: Main features which can be taken into account while evaluating the recommender system

where m is maximal rank. By solving optimization problem we obtain the best vector

$$\hat{r}_i = p_i \sqrt{\frac{(1/\alpha^2 - 1)\sum r_j^2}{\sum p_j^2}} + \frac{c_i}{\alpha}. \tag{2.19}$$

We can easily derive the expected profit for this vector

$$p \cdot \hat{\phi}(\hat{r}) = \frac{1}{m}\left(\sum_i p_i^2 \sqrt{(1/\alpha^2 - 1)\sum r_j^2 / \sum p_j^2} + p_i r_i/\alpha\right), \tag{2.20}$$

which can be simplified by the Cauchy-Schwarz inequality [37] to

$$p \cdot \hat{\phi}(\hat{r}) \geq \left(\sqrt{\frac{1}{\alpha^2} - 1} + \frac{1}{\alpha}\right) \frac{\sum p_i r_i}{m}. \tag{2.21}$$

The expected profit from r is just $\sum p_i r_i/m$, thus the expected gain is equal to at least

$$\frac{p \cdot \hat{\phi}(\hat{r}) - p \cdot \hat{\phi}(r)}{p \cdot \hat{\phi}(r)} = \sqrt{\frac{1}{\alpha^2} - 1} + \frac{1}{\alpha} - 1 \geq 2(1/\alpha - 1). \tag{2.22}$$

This means that when derivation from the best recommendations at the level of 10%, i.e. $\alpha = \frac{9}{10}$ is allowed, then, the company's profit increases by at least 22%. More detailed examples with weaker assumptions can be found in [27].

Chapter 3

Data mining and statistical methods

In this chapter, I present most popular data mining and statistical models used in recommender systems. First, distance and similarity measures are developed. Then, I discuss data preparation including dimensionality reduction. At the end classification and regression methods are presented. With each method I try to discuss cases in which it is useful in recommender system.

3.1. Data preparation

Usually in real-life problems we deal with chaotic datasets with too many attributes and outliers which would bias results if left untouched. Several preprocessing methods are employed depending on the application.

3.2. Sampling

One of the main techniques for selecting subsets of data in data mining is called sampling. There are several reasons to sample from dataset - it might be used for preprocessing and for data interpretation as well. Idea is also used for reducing size of data when computation is too expensive [2].

Another common usage is evaluation process where we split data into training and test sets. The rule of the thumb is to split in proportion 80/20, however it depends on application.

Improper approach to sampling may bias our model towards unexpected areas. To deal with this problem usually *random sampling* or *stratified sampling* are employed, where the later one splits data into portion according to selected features and follow it with random selection.

3.3. Dimensionality reduction

While developing RS it is common that we deal with high dimensional space and very sparse information. In this setup distances and similarity between users or items become less meaningful. The problem is referred as the Course of Dimensionality. There are several methods available and here I present two most common: Principal Component Analysis and Singular Value Decomposition.

3.3.1. Principal component analysis

The idea was successfully applied in many recommendation algorithms [43] - the direct implementation was developed by Manolis G. Vozalis *et al.* [10].

Definition 7. *PCA is orthogonal linear transformation of dataset* $\mathbb{X}$ *into new coordinate system such that greatest variance lies on the first coordinate (first principal component), the second greatest variance on the second coordinate, and so on.*

It turns out that there is direct equation for this transformation, provided we know eigenvectors of the matrix $\mathbb{X}$.

Theorem 8. *Let* W *be the matrix of eigenvectors of* $XX^\top$ *organized in the descending order of corresponding eigenvalues. PCA transformation of* X *is given by* $Y = W^\top X$.

Proof. Without loss of generality we can assume that X has mean zero. Let $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ be eigenvalues of $XX^\top$ and v_i corresponding orthonormal base. Assume that there is $w = \sum_{i=1}^n a_i v_i$ which maximize the varience and $w \neq v_1$. Then

$$\begin{aligned} w_1 &= \text{argmax}_{\|w\|=1}\, w^\top XX^\top w \\ &= \text{argmax}_{\|w\|=1} \left(\sum a_i v_i\right)^\top XX^\top \left(\sum a_i v_i\right) \\ &= \text{argmax}_{\|w\|=1} \left(\sum a_i v_i\right)^\top \left(\sum a_i \lambda_i v_i\right) \\ &= \text{argmax}_{\|w\|=1} \sum a_i^2, \end{aligned}$$

hence w_1 maximize the variance iff $w_1 = v_1$. With $k-1$ principal components, the kth one can be found the same way after following transformation

$$X_{k-1} = X - \sum_{i=1}^{k-1} w_i w_i^\top X$$

and substituting X with X_{k-1}

$$w_k = \text{argmax}_{\|w\|=1}\, w^\top X_{k-1} X_{k-1}^\top w.$$

□

This theorem gives useful interpretation of the PCA, however it is not efficient to find eigenvectors. Another approach provides constructive process to find transformation

Let me assume X has zero mean. Our goal is to find an orthonormal transformation P such that PX is a random vector with pairwise uncorrelated components.

$$\begin{aligned} \text{cov}(PX) &= \mathbb{E}[PX\ (PX)^\top] \\ &= \mathbb{E}[PX\ X^\top P^\top] \\ &= P\ \mathbb{E}[XX^\top] P^\top \\ &= P\ \text{cov}(X) P^{-1} \end{aligned}$$

Since $\text{cov}(X)$ is non-negative definite matrix, there exists P which satisfies the equation.

This process is constructive, however it is still not efficient for huge matrices. One of simple algorithms used in practice takes advantage of convexity. Consider following pseudocode

p {random vector}
repeat
  $t = 0$ {vector}
  for all rows $x \in X^T$ **do**
    $t \leftarrow t + (xp)x$
  end for
  $p \leftarrow t/|t|$
  $i \leftarrow i + 1$
until *stop_condition*
return p

where c is constant number of steps derived empirically optimizing performance. In this approach we basically calculate $X^T Xp$, normalize and assign the result again to p. This process converges to the first principal component. In order to find others we can just subtract p from X^T like in Gram-Schmidt orthogonalization [3].

3.3.2. Singular Value Decomposition

Ideas of using linear algebra for information retrieval where summarized in [11]. The goal of this section is to derive an approximation of data matrix X by low-rank matrix $\hat{X}$. In case of recommender systems we can think of the rank as the number of user's personality types.

We define singular value decomposition of an $m \times n$ matrix X as a factorization

$$X = U\Sigma V^*$$

such that where U is an $m \times m$ matrix, Σ is an $m \times n$ nonnegative diagonal matrix, and V^* is an $n \times n$ unitary matrix. We refer to the diagonal entries of Σ as the singular values of X, and to the m columns of U and the n columns of V as left singular vectors and right singular vectors of M, respectively.

One of the direct applications of SVD in recommender systems is approximation of an X matrix with low rank $\hat{X}$. Let me start with following definition

Definition 9 (Frobenius norm)**.** *Let $m \times n$ matrix A,*

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{\operatorname{tr} A^*A}$$

is called Frobenius norm.

In year 1936 Eckart and Young developed following method of low-rank matrix approximation in this norm

Theorem 10 (Eckart–Young theorem)**.** *Let X be the $m \times n$ matrix with SVD decomposition $X = U\Sigma V^*$ and let $r \in \mathbb{N}_+$. Then*

$$\hat{X} = U\hat{\Sigma}V^*,$$

where $\hat{\Sigma}$ is as Σ except it has only r highest singular values, maximizes the Frobenius norm among all matrixes of rank r.

Proof. We want to minimize $\|X - \hat{X}\|_F$ with constraint $\operatorname{rank}(\hat{m}) = r$. From the definition of Frovenius norm we can easily see that it is invariant to unitary transforms so we can express the task as minimization of

$$\|\Sigma - U^*\hat{X}V\|_F.$$

Since Σ is diagonal, in order to minimize the norm we expect $U^*\hat{X}V$ to be diagonal as well. Thus U and V are singular matrices not only of X but also of $\hat{X}$. We can now express $\hat{X}$ as $\hat{X} = USV^*$ and then our optimization task simplifies to finding S minimizing

$$\|\Sigma - S\|_F,$$

but S has maximum of r non-zero entities so choosing the highest r from Σ minimizes the error. □

In order to compute SVD of a matrix X one can follow two-step procedure:

1. reducing to bidiagonal matrix,
2. computing SVD of the bidiagonal matrix.

Although there are direct methods for computing SVD of the bidiagonal matrix, in practice ϵ-approximation methods are employed which with given constant ϵ can be performed in $O(n)$ steps. The more expensive part is the decomposition in first step which is usually done by Householder reflections in $O(mn^2)$ steps (assuming $m > n$).

In many data mining algorithms SVD is fundamental, however, it has two principal shortages [11]:

- quadratic complexity,
- need of random access to complete data sets.

It leads two high resource consumption and in some cases might be ineffective.

Freize *et al.* [38] proposed the following way to approximate the SVD of an matrix X. SVD of $XX^\top$ can be written in terms of X's SVD as $XX^\top = U\Sigma^2U^T$, thus the SVD of X can be derived from that of $XX^\top$ and vice versa. Now choose a subset of $XX^\top$ columns and form the matrix S from them. It turns out that SVD of $SS^\top$ converges fast to the $XX^\top$'s one and we can derive from it an approximation of X decomposition. SVD on S takes only $O(s^2m)$ (assuming $s << m$), thus this method gives a fast approximation.

As described at the beginning of this section, SVD is commonly used to derive relations between users and products. However Sarwar *et al.* [39] proposed this approach also as a precomputation for later use in a kNN approach. They also mentioned that one of the biggest advantages of SVD is the existence of incremental algorithms what allows on-line approaches.

Since SVD provides a basis for many collaborative filtering algorithms, it will be discussed in details in chapter 4.

3.4. Clustering methods

Clustering, also referred to as classification, is the methodology of spiting data into subsets by comparison of its features. There are dozens of different methods and aims, however usually a goal of this process is finding groups of datapoints which are far from each other in some sense. As measure of distance we can take various space norms and similarity measures.

We will talk about either *supervised* or *unsupervised* methods. First one assumes that classes are known and we can provide training dataset with right classification, while the later tries to divide data in logical sections using only data features.

In case of recommender systems we might deal with both cases. Supervised approach helps for instance in news recommender system when some users specified their interests and

some did not. Unsupervised classification may be employed when we want to split items into groups which would attract the user altogether.

The later case leads to the need of multi-classification which will be also introduced in chapter 3.5.

One can also think of the recommendation process as of the supervised learning itself - basing on users features we try to predict if user likes the item. In this chapter I present the most popular methods, successfully applied by Ungar and Foster in 1998 [6].

3.4.1. k Nearest Neighbours

One of the simplest, fastest and easiest to implement and thus quite frequently used method is k Nearest Neighbours. It does not build a model, that is why it is commonly referred as *lazy learner*.

As the name suggests, the idea is to check class of k nearest neighbours and classify it according to the majority.

Despite it's simplicity, the kNN method shows good accuracy with very short running time and has been challenge only by methods which use low-rank matrix approximation. It constitutes *de facto* standard for collaborative filtering [1].

3.4.2. k-means

While the kNN works good in *supervised* case in it's standard version it does not help in *unsupervised* problem. One of the similar approaches in the later task is to form classes using following algorithm

S {set of points to classify}
$C_i \leftarrow \{p_i\}$ {where are $p_i \in S$ pairwise different}
repeat
  for all C_i **do**
    $m_i \leftarrow \text{mean}(C_i)$
    $C_i \leftarrow \emptyset$
  end for
  for all $p \in S$ **do**
    $i \leftarrow \text{argmin}_i \, \text{dist}(m_i, p)$
    $C_i \leftarrow C_i \cup \{p\}$
  end for
until *stop_condition*

Where *stop_condition* might be first not-changing, expected accuracy, fixed number of steps or others.

3.4.3. Artificial Neural Networks

Another classification method, called Artificial Neural Network, is based on human's brain behaviour. It is wildly used in data mining however, there is no conclusive study to whether ANN introduce any performance gain [2].

In basic binary setup we have k layers, in each layer $i < k$ we have n_i neurons and 1 output neuron at the end. Each neuron has inputs from neurons in preceeding layer and outputs to the further ones.

We can think of first layer as the features. Classification process goes through all layers iteratively. Let me denote the value of ith neuron in jth layer as $v_{i,j}$. First we populate first

layer $v_{i,1}$ with features values and then proceed iteratively so that

$$v_{i,j} = \begin{cases} 1 & \text{if } \sum_{i=1}^{n_{j-1}} w_{i,j} v_{i,j-1} \geq \theta_{i,j}, \\ 0 & \text{if } \sum_{i=1}^{n_{j-1}} w_{i,j} v_{i,j-1} < \theta_{i,j}. \end{cases}$$

Where $w_{i,j}$ and $\theta_{i,j}$ are learned parameters.

The main advantage of ANN is that they can cover dependence more complex than linear. Number of layers corresponds to the degree of polynomial which can be reproduced by the network. Theoretically each continuous function can approximated with high order ANN, however one should be aware of over-fitting. In practice changing the number of layers is a parameter for controlling trade-off between generality and fitting.

There have been many methods of learning discovered. Although there were already few tries for tackling recommender system with use of ANN [35][36], general accuracy of ANN methods in making suggestions is still an open question.

3.4.4. Support Vector Machine

The goal of Support Vector Machine (SVM) is to split the space into hyperplanes such that margin of border is maximized. This approach aims to minimize possibility of misclassification marginal instances in future.

Formally we define linear separation by hyperplane

$$\langle w, x \rangle + b = 0.$$

Then classifier

$$f(x) = \begin{cases} 1 & \text{if } \langle w, x \rangle + b \geq 1, \\ -1 & \text{if } \langle w, x \rangle + b < 1. \end{cases}$$

In two dimensional case we can define our problem by providing a function

$$L(w) = \frac{\|w\|^2}{2}$$

which we will try to minimize.

We use SVM for both *unsupervised* and *supervised* case. In the later one the separating hyperplane might not exist at all. There are several methods to deal with this problem. We may provide so called *soft margin* by adding penalties for misclassification. Other popular method is *kernel method* which transforms the space before applying SVM.

As an example let me take points

$$p_1 = (1, 1), p_2 = (-1, -1), p_3 = (-1, 1), p_4 = (-1, 1)$$

and classes $A = \{p_1, p_2\}$ and $B = \{p_3, p_4\}$. There is no line which splits the space into classes however when we are given kernel function

$$\Phi(x_1, x_2) = (x_1, x_1 x_2, x_2)$$

transformed points are easily splittable by the plane $P = \{(x, 0, z) : x, z \in \mathbb{R}\}$.

In all cases SVM can be treated as an optimization problem. This leads to exploitation of similar implementations. One commonly used in practice is based on Newton-like iterations for solving Karush–Kuhn–Tucker conditions. To avoid problems with large equation systems, low-rank matrix approximation is also employed.

Another approach is Platt's [40] Sequential Minimal Optimization (SMO) algorithm, which divides the problem into 2-dimensional subproblems for which analytical expression is available. This approach eliminates the problem of numerical approximation.

3.5. Multinomial logistic regression

In some more sophisticated recommender systems one aim to predict not only how does the item suit given user but also how does it fit in given context. Imagine that one is aiming to choose the best restaurant and define the context as some set of tags, for instance "business lunch", "date" or "beer with friends". Of course one can use standard logistic regression for each tag, however, in this case data enclosed in covariance of tags may be lost. That is base motivation for using Multinomial logistic regression. Applications can be found in [4].

Chapter 4

Techniques

Recommendation systems can be classified as content-based and user-based. Methods based on the content exploit data contained in item details and eventually data provided by experts. User-based systems pay attention only to iterations among users and items. A priori each item is treated equally. It is based on the assumption "if someone who ranked some set of items similar to me and he like item which I do not know, then probably I would like that item as well". Usually mixing both approaches gives the best results.

Throughout this chapter both approaches will be discussed in details.

4.1. Content-based methods

The general goal of this approach is to characterize items which user likes and suggest him those which share this characteristic. Information about an object i might be stored in the feature vector $\mathbf{x}_i$. For data in the form of text documents, like websites or reviews, vectors containing frequency of occurrence are commonly used. On the other hand, for each user u we compute his preference profile vector $\mathbf{x}_u$. Common technique, used in such system as Newsweeder [52], is to update the profile $\mathbf{x}_u$, whenever user ranks item i by the x_i in proportion of the rank $r_u i$, i.e.:

$$x_u = \sum_{i \in I_u} r_{ui} x_i, \tag{4.1}$$

where I_u describes the set of ranked items.

Then recommender can for instance suggest items which are similar to user's taste according to cosine similarity or Minimum Description Length (MDL) [52]. Similarly we can use this for rating prediction or classification, by building for each rating value $r \in S$ content vector as an average of feature vectors of items which received rating r from given user. Then predicted rating is the value corresponding to ranking vector most similar to the item vector.

This approach suffers mainly from problems like over-specialization or limited content analysis (i.e. too high cost of item's annotations). Another problem is the mapping from item space to vector space - for instance it may be hard to distinguish two articles when they use similar words.

4.2. Collaborative filtering

Basic idea behind the collaborative filtering states that if ratings of users u and v are similar, and user u rated item i which user v did not rank, then probably v's rank would be similar

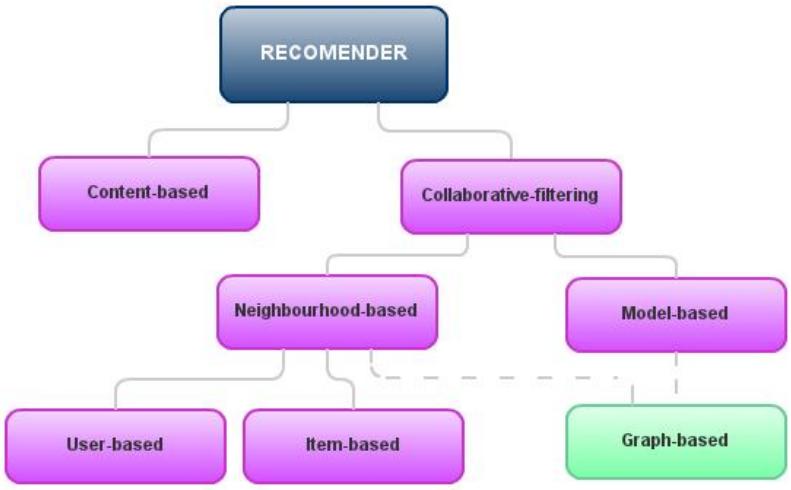

Figure 4.1: Hierarchy of recommender system methods. Graph-based techniques may be treated as an model or as neighbourhood-based predictions as well.

to the one that u gave. Although the assumption seems to be simple, variety definitions of similarity lead to various methods and evaluations.

Collaborative approaches deal with some limitations of content-based methods. No expert knowledge is required, since suggestions are based only on ratings. Additionally specialization is some particular area is avoided, because here the system is able to recommend items with very different content. In case of text data another important thing is not relying on the content itself as an indicator of quality.

Those methods can be grouped in two general classes *neighbourhood-based* and *model-based*. In the first one user's ratings are used directly to predict new ones. This is done in two ways (as the hierarchy diagram 4.1 shows), by *user-based* or *item-based* methods. User-based looks for user's with similar ratings, while item-based predict ratings by checking users ratings on similar items.

In opposite to neighbourhood based methods, which predict directly from the ratings, model-based use ratings to build a statistical model. General idea is to derive from user and item data latent characteristics.

4.2.1. Neighbourhood-based vs Model-based methods

Recent research on the recommender systems shows that model-based methods predict better than the neighbourhood based ones [46][47], it is also well-known that prediction quality does not imply user's satisfaction [42].

Very important threat for model-based approach is that they may lead to over-specialization. They are great in retrieving information from users and items however they also retain in the same taste area, while serendipity is key factor for many users.

Next advantage of neighbourhood methods over model based one is simplicity. They are relatively easy to implement and control, what leads to possibility of testing many approaches. Additionally generated result is easier to justify and in some systems this is also a requirement. Moreover simplicity leads also to robustness and the same code may be used in different domains, since predictions are based solely on the ratings.

4.2.2. User-based prediction

Let $\mathcal{N}(u)$ neighbours (users with similar tastes) of user u and $\mathcal{N}_i(u) \subset \mathcal{N}(u)$ users who ranked item i. Basic user-based method to predict unknown user's u rating of item i is to calculate:

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} r_{vi} s_{uv}}{\sum_{v \in \mathcal{N}_i(u)} |s_{uv}|}, \tag{4.2}$$

where s_{uv} denote similarity of users' ratings.

Similarity measures

Methods of defining similarity between users are derived from standard statistical analysis. We may use, among many others, the *cosine similarity*:

$$cos(\mathbf{x}_u, \mathbf{x}_v) = \frac{\mathbf{x}_u^T \mathbf{x}_v}{||\mathbf{x}_u|| ||\mathbf{x}_v||},$$

with putting zeros on not ranked entities. Another measure is the *Pearson's Correlation* defined as:

$$PC(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2 \sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}$$

where I_{uv} is the set of items reviewed jointly by both users. PS measure tries to catch likes and dislikes by comparing rating value with users average $\bar{r}_u$. *Spearman Rank Correlation* is applied as well. Another interesting similarity measure employed in studies is *Mean Square Difference*:

$$MSD(u,v) = \frac{1}{|I_{uv}|} \sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2.$$

The main limitation of this measure is that it does not capture negative correlation, which may be useful in some cases.

Another problem with similarity is that there are items which almost everyone prefers like for instance "The Godfather". Relying on them may bias the system unpredictable. One way to deal with this problem is to weight the values by item's popularity, which in case of *Pearson's Correlation* lead to *Frequency-Weighted Pearson's Correlation*

$$PC(u,v) = \frac{\sum_{i \in I_{uv}} \lambda_i (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} \lambda_i (r_{ui} - \bar{r}_u)^2 \sum_{i \in I_{uv}} \lambda_i (r_{vi} - \bar{r}_v)^2}},$$

where $\lambda_i = \log(\frac{U}{U_i})$, U is the number of users and U_i is the number of users who ranked item i. Analogical process results in the measure for item-based systems.

Similarity measure may be treated as easily pluggable component of neighbourhood-based system. We can test among many functions independently to other parts and choose the one which performs the best.

4.2.3. User-based classification

Previously calculated values are almost surely non-integers. This is not a problem when we are going to order items afterwards, however sometimes showing the predicted number to the user is required and moreover the integer is preferred. Instead of mapping predicted floating point number to the nearest integer, user-based classification is usually employed.

Let me define for each possible rating $r \in \mathcal{S}$ the following value

$$z_{uir} = \sum_{v \in \mathcal{N}_i(u)} \delta(r_{vi} = r) s_{uv}, \tag{4.3}$$

where δ is 1 if $r_{uvi} = r$ and 0 otherwise. Then predicted class is just an r with highest z_{uir}.

4.2.4. Item-based method

Here instead of looking for similar users we look for similar items. So corresponding prediction is defined by following equation

$$\hat{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_u(i)} r_{uj} s_{ij}}{\sum_{j \in \mathcal{N}_u(i)} |s_{ij}|}, \tag{4.4}$$

where $\mathcal{N}_u(i)$ denotes neighbourhood of item i restricted to those ranked by user u and s_{ij} denotes similarity of items. Classifier is defined similarly by the r with the highest

$$z_{uir} = \sum_{j \in \mathcal{N}_u(i)} \delta(r_{uj} = r) s_{ij}. \tag{4.5}$$

4.2.5. Comparison of user-based and item-based

According to Christian Desrosiers *et al.* [48] there are 5 major factors which should be taken into account while choosing between item-based or user-based methods.

- **Accuracy** - In this case it depends mainly on the ratio between users and items in the portal. Basically if we have lots of users and not many items the item-based approach is preferable (like in *Amazon.com*) and vice versa.

- **Efficiency** - Similarly the complexity depends on the ratio between users and items. Let me define $p = R/U$ and $q = R/I$ where R, U, I are numbers of ratings, users and items. Than the complexity of user-based method is proportional to p^2/I while the complexity of item-based one is proportional to q^2/U

- **Stability** - We should take into account what increases faster in our system - the number users or items. If the set of items is rather static than item-based learns faster and vice versa.

- **Justifiability** - One advantage of item-based recommendations in some systems is it's justifiability. Because of privacy reasons we may not be able to show similar users who reviewed the item, while showing similar items is usually not a problem at all.

- **Serendipity** - While optimizing this factor item-based method is less preferable, because the rating of an item is predicted by use of ratings of similar items. It leads to safe recommends but on the other hand user is less likely to result in something surprisingly interesting. In the user-based approach, except of users similarities we gain also knowledge of areas already unchecked by given user.

4.2.6. Matrix completion

Another, relatively successful approach is trying to define a goal in strictly mathematical way and apply well-known solvers. We can think of the rating prediction as of the matrix completion problem. While having sparse matrix of users' ratings M_0 we may ask how to complete it to $\hat{M}$ so that we approximate the real unknown one M as good as we can. This is where mathematical analysis from chapter 3 comes into play.

The idea is to approximate M in the Frobenius norm by the low-rank matrix. We can think of low-rank as the way of generalization, so that we will not overfit the model. Unfortunately SVD method proposed in section 3 can not be applied directly, since we have unknown values in the matrix. Tries of substituting missing values with average would bias very much, because after this operation our matrix becomes almost-constant.

Common solution to this problem is to define $P = U_k\Sigma_k^{1/2}$ and $Q = V_k\Sigma_k^{1/2}$ and learn them only using known ratings [47], i.e. we minimize the following error function

$$err(P,Q) = \sum_{r_{ui}\in R} (r_{ui} - p_u q_i^T)^2 + \lambda(||p_u||^2 + ||q_i||^2),$$

where λ is the parameter which controls regularization level. Similar approach may be employed in a neighbourhood-based recommender system [49] by solving:

$$err(P,Q) = \sum_{r_{ui}\in R} (z_{ui} - p_u q_i^T)^2,$$

with constrains

$$||p_u|| = 1 \forall_{u\in U},$$
$$||q_i|| = 1 \forall_{i\in I},$$

where z_{ui} is the rating r_{ui} mean-centered and normalized to $[-1, 1]$.

4.2.7. Netflix prize

The Netflix Prize was an open contest for collaborative filtering algorithms which predicts user ratings for movies. The goal of competition was to improve former algorithm by the 10% of evaluation function value. As performance measure Root Mean Square Error was selected.

Chellange was organized by an on-line DVD-rental company Netflix, which provided prize of US$1,000,000. Netflix published a training data set of 100,480,507 ratings of 480,189 and 17,770 movies with corresponding date of vote. Each rating was an integer from set 1 to 5.

High price and huge data set attracted many scientists and prize-hunters to develop new techniques. Due to the lawsuits Netflix dataset is no longer available on the Internet. However, scientists from University of Minnesota (GroupLens) set up the project of movie recommendations and provided publicly legible dataset of 10million ratings, which enabled further research.

Winner of each phase were obliged to publish a paper with resulting algorithm. From the winning Bellkor team [43], we can learn various well-applied sophisticated methods with conclusion that the weighted sum of not-fully-correlated algorithms performs the best.

Netflix prize had tremendous impact on understanding and exploiting collaborative-filtering. In the winning algorithm even day of the week on which the review was posted was taken into consideration. Each additional information may help to build better system, but privacy issues always need to be considered. Although Netflix announced the sequel of

competition, it was later dismissed due to possibilities of privacy abuse [50]. Because of the same reasons dataset is no longer available on-line - fortunately MovieLens took over Netflix place in scientific community.

Napoleon Dynamite's problem

In the interview [45] members of winning teams describe as one of main problems prediction of ratings on movies with high variance. Since Netflix price the problem is often referred to as a Napoleon Dynamite's problem [45]. According to Bellkor's statement, the problem is still open and there is no reliable method known yet.

Measure controversy

Apart from privacy issues, another thing which rose controversies is the RMSE measure as the only performance measure. Participants doubted if improving it by 10% has really so important impact on the users' satisfaction [51] or if it is better to adjust other factors, like trust or serendipity. On the other hand this improvement is sufficient to result in changes in user's personal TOP10. Additionally it only gives extra information which do not need to be exploited directly - it might be just a complement for other methods.

4.2.8. Graph-based methods

Among more sophisticated model and neighbourhood methods, one of which started to gain popularity lately is based on perceiving relation between users and items as an bipartite graph.

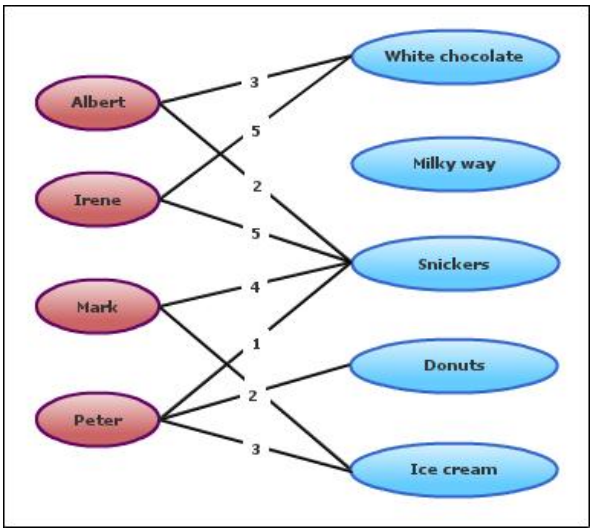

Figure 4.2: Graph representing users' preferences

We can treat edges of the graph as indicators of correlation, where the straight of relation is proportional to the value. By going through edges we can predict correlation with unrated items. This process is commonly referred to as a propagation.

Another approach is to perform random walk with probabilities proportional to edge values. Finally we can also try to employ algorithms searching shortest path. Detailed information about similar approaches can be found in *neighbourhood-based*.

4.3. Summary - case study: sweet recommender system

In this section I tried to evaluate simple implementations of described methods. Despite the existence of large datasets and algorithms adjusted to them I decided not to rely on them. Afraid of disability of generalization of methods developed on movie datasets I created small open-source project where users rate sweets and put it on-line under the address http://sweetrs.org/. Features of them are relatively easy to describe, so it gave me enough data for neighbourhood-based and content-based approaches. Data already submitted contains almost 30000 ratings. Purpose of this study was testing different approaches and considering basic relations between them, not to build the best possible algorithm.

Figure 4.3: sweetrs.org: rating site

Users were asked to rate sweets on the 1-5 integer scale. If one does not know given item he was asked to click "Never tried" button. I aimed to produce the matrix as dense as possible so majority of sweets were popular ones and as a result there were only 7% empty entries.

I applied four algorithms:

1. Content-based,
2. User-based,
3. Item-based,
4. Low-rank matrix completion,

with several similarity measures and parameters. Additionally I created 2 predictions based on weighted sum of those algorithms. As a benchmark algorithm simple mean was used. All algorithms were written in matlab (source is attached in the appendix) and Matrix completion was performed by Iterated Soft Thresholding algorithm provided by Majumdar A. [53]. As an error measure the RMSE was used.

Results depending on the split between test and training set are presented in table 4.1 and on the plot 4.4.

Matrix completion, item- and user-based approaches were applied as described previously, without additional knowledge. For content-based approach I used additional table with description of products' properties: chocolate, cocos, caramel, ice, sugar, jelly, mint, nuts, honey, nougat, fruit, chewy, lozenge, white chocolate, crunchy, waffle and cream. I tried to

express my personal experience about importance of given property in the scale $0 - 1$. As a content-based rating classification method, the one proposed in section 4.1 was applied.

As we can see on the results this approach was outperformed by all methods including simple mean algorithm. First of all, problem comes from the fragile structure of the content-based classification. Although it grasps preferences nicely, it is hard to distinguish how do they map to ratings afterwards. Secondly if one likes chocolate and mint, it does not necessarily imply that he likes mint with chocolate (like After Eight© chocolates).

Apparently the best outcome may be achieved by joining several models - in this case weighted sum of user-based, item-based approach and the mean.

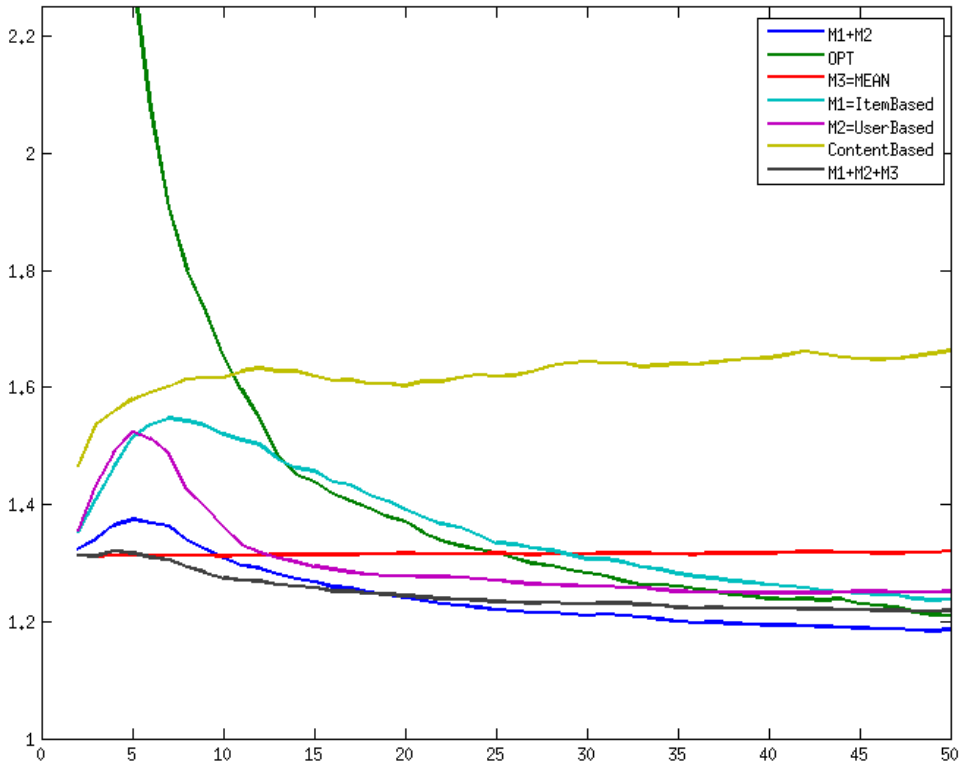

Figure 4.4: Comparison of applied algorithms. Horizontal axis denotes the fraction of dataset used as training set (in %) and vertical one indicates RMSE. Till the 18% train/test threshold, joined model of user- and item-based approach and the mean appears to be the best performing algorithm. After that, joined model without mean outperforms the previous one.

When ration of training dataset is huge (over half of the whole set), matrix-completion algorithm outperforms others. Than when weighted with M1 and M2, the sum achieved over 13% improvement over the constant mean. However the assumption of training dataset of over 50% is unrealistic. That is why the joined model of user- and item-based algorithms may be considered as the best performing approach among proposed.

Although the experiment was conducted in the Netflix fashion, there are some key differences important to mention:

- Error of the algorithm which assigns mean to all unknown entities performed worse in

	1%	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
M1 + M2	1.31	1.37	1.30	1.26	**1.24**	**1.21**	**1.21**	**1.20**	**1.19**	**1.18**	**1.18**
Opt	1.64	2.31	1.65	1.43	1.37	1.31	1.28	1.26	1.23	1.23	1.21
M3 = Mean	1.31	**1.31**	1.31	1.31	1.31	1.31	1.31	1.31	1.31	1.31	1.32
M1 = ItemBased	1.32	1.51	1.51	1.45	1.39	1.33	1.30	1.28	1.26	1.24	1.23
M2 = UserBased	1.32	1.52	1.36	1.29	1.27	1.27	1.25	1.25	1.24	1.25	1.25
ContentBased	1.39	1.58	1.61	1.61	1.60	1.61	1.64	1.64	1.65	1.64	1.66
M1 + M2 + M3	**1.31**	1.31	**1.27**	**1.25**	1.24	1.23	1.23	1.22	1.22	1.21	1.21

Table 4.1: Results of the experiment. Columns describe RMSE in given train/test setup (label is percentage of data used for training). Bold font indicates best performing algorithm.

my experiment ($RMSE = 1.31$) than in Netflix (1.0540). This indicates much higher variance of data in my setup.

- Netflix dataset is very sparse (1.5% known entities) in comparison to mine (over 93% known entities). In Netflix dataset users rate only items which they have watched so set of rated items correlates with tastes. In my dataset training entities are chosen at random, so this correlation is lost. This also results in uniform distribution of empty grades among users and items so simplifies the algorithms.

Error of the best model is about 10% smaller than the mean algorithm. This should result in relatively good TOP5 of unrated items which is usually goal of similar procedures.

Chapter 5

Case study: Building the recommender system for neib.org

Let me introduce a real-life example of an application, which indicated the direction of this thesis. Neib.org is the Central European portal with user's reviews on local venues. The project was started under codename dood.pl in 2009 by two IT master students of University of Warsaw - Paweł Bedyński and myself.

Since then we gathered over 10000 reviews of over 50000 polish companies. With its daily unique visitors number at the level of 2000, the website constitutes one of main sources of local venue information in more than 15 polish cities.

In the year 2010 Paweł Bedyński, proposed advertising model in his bachelor thesis on economics faculty. In the same year an iPhone application which I built for Neib.org, constituted major part of my master thesis on computer science faculty. This year extended analysis of Andoroid application built by Paweł Bedyński, become his master thesis on computer science faculty at the University of Warsaw.

Project is officially founded by my own company LEMONET, however because of it's non-commercial nature, it attracted many students to contribute - over 50 internship position was successfully taken by students from various polish university faculties.

In this chapter I try to apply already presented methods in the real problem. I start with definition of goals in general way, with formal mathematical definition of results evaluation afterwards. Then I describe factors indicating set of methods which I should test. Then I apply the methods and describe related technical details. Finally, I focus on evaluation of proposed methods and I try to drawback some conclusions.

5.1. Main goals

Problem which I tried to solve, naturally arose from analysis of the website's statistics. According to Google Analytics© software, which we use throughout Neib.org, one of main problems is that vast number of users leaves our site just after getting in. In e-commerce terminology this measure is called Bounce rate (BR) and is defined as follows

$$BR = \frac{LV}{AL},$$

where AL are all visitors and LV is the number of visitors which leave the website in less than 10 seconds. In the year 2010 bounce rate of the website was equal to 85%. Since our content

might be consider attractive, as we have relatively vast amount of reviews (in comparison to concurrent websites), it indicated that navigation to valuable content may be the problem.

Over 90% of our users come from the search engines, while 85% of landing pages are the sites of particular businesses (figure 5.1). Bounce rate on those landing pages is also at the level of 85%. From people who enter through search engine vast majority typed keywords related with name of a venue (not a category).

On the other hand studies shows [17] that the more user interact with a site the more likely he is to register, what we consider as the goal of current stage of website's progress.

This simple analysis leads us to conclusion that we may gain the biggest number of new user by trying to keep on site longer those, who entered by one of business details pages.

The way to achieve this goal, proposed in this chapter is based on following simply-posed hypothesis

Hypothesis 11. *Bounce rate on the business details page is significantly negatively correlated with the attractiveness of related outgoing links specified on figure 5.1*

In the following chapter I will prove it directly by developing algorithm which decreases BR by improving quality of links.

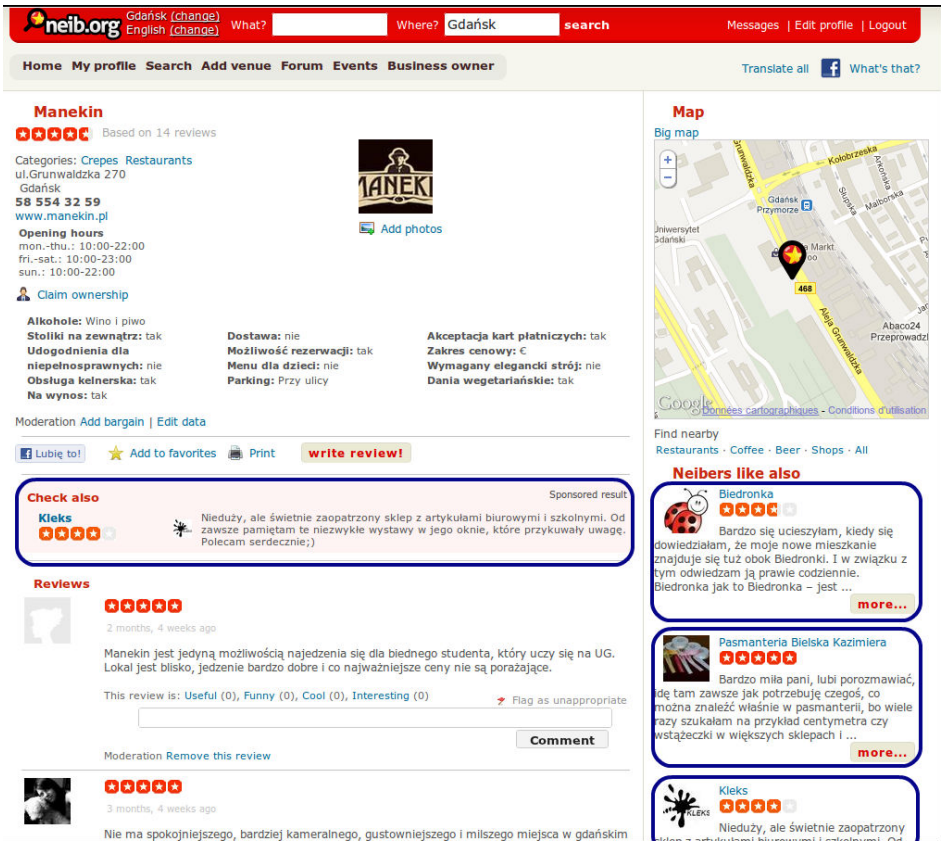

Figure 5.1: slots to optimize

5.2. Evaluation methods

Keeping in mind the last subsection and chapter 2, I try to develop measure for evaluation of methods. In order to ensure accuracy, two functions will be used in the study. The first

one evaluates directly the quality of links as their Click Through Rate (percent of users who clicked a link after seeing it).

Definition 12.

$$CTR = \frac{clicks}{views}$$

The second takes into account the goal itself i.e. the bounce rate.

Fortunately Google Analytics provides tool, which helped to mine historical values of those measures. Since all business details pages have URI starting with `/biz/` we can easily filter and check BR among them. Similarly CTR on boxes can be measured as ratio of pages `/biz/` followed by `/biz/` on the user's navigation path (figure 5.2).

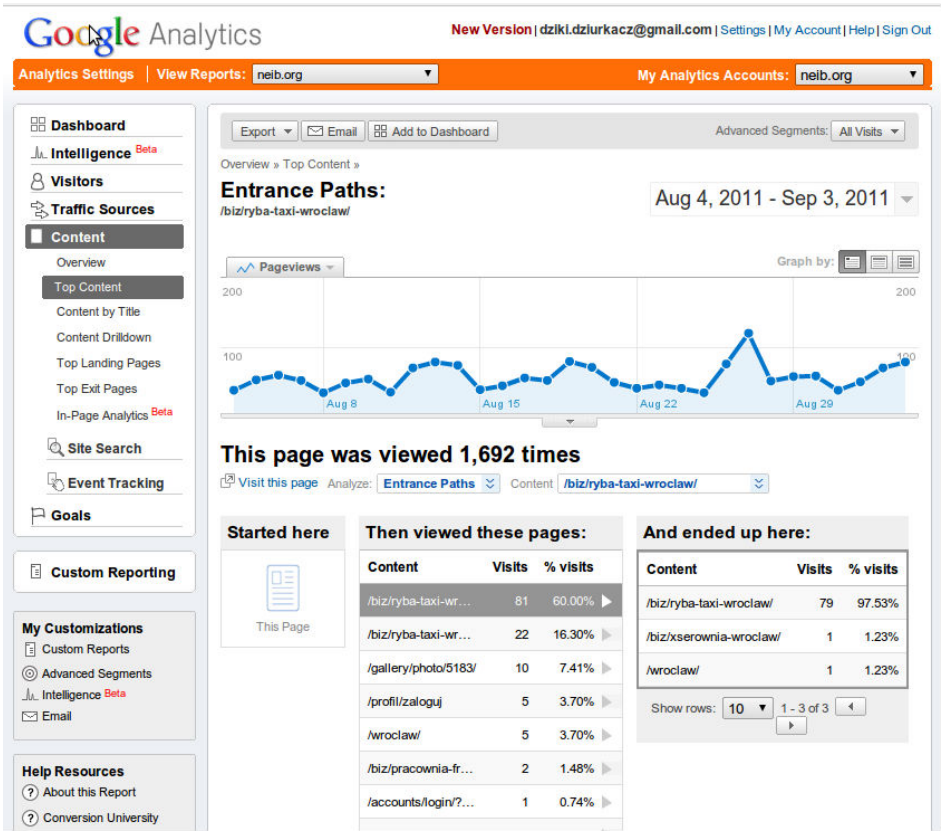

Figure 5.2: tracking CTR on considered slots

5.3. Input & output

Let me define input variables into two disjoint groups: everything we already know about businesses (content) and everything we know about the user (context). Content group contains among others:

1. business description, category, photo, etc.,
2. overall business rating,
3. reviews,

4. average time spent on given site.

Context contains:

1. user's last page,
2. search phrases from web-search if avail,
3. new or returning visitor,
4. average time spent on given site.

Using this data we aim to pick optimal 4 venues, which given user is most likely to click on (i.e. which maximize evaluation function).

5.4. Recommendation methods

Before implementation of algorithms described in following section, venues displayed in the slots where randomized. Algorithm can be simply described as follows:

1. Consider only business from the same city,
2. If there are reviewed venues from categories of current one, then pick 4 of them uniformly at random,
3. Otherwise pick 4 of all reviewed businesses from given city.

Since there is to little data for performing reliable data mining and collaborative filtering, I applied paradigms of Content-based Information Filtering (IF). Most of the users are not even logged in, so it is hard to adjust the system to their context, if we do not have history.

I tried to exploit design suggested by [16] presented on figure 5.3. In IF approach system needs proper schema for representing items and profiles. The recommendation is performed in the following three steps, however in this work I focus mainly on the first one. For each step corresponding component needs to be implemented:

- **CONTENT ANALYZER** - responsible for preprocessing of an item. It concerns such ideas as information retrieval from text comments, aggregation of ratings, finding relations between items. Representation generated by this component is input for the PROFILE LEARNER and the RECOMMENDER.
- **PROFILE LEARNER** - task of this module is to retrieve relations between users and summarize information which we have about the current one. After each recommendation we can store positive and negative feedback, so to apply machine learning techniques in order to know which kind of suggestions are most efficient.
- **RECOMMENDER** - this module summarize all information already gathered and tries to predict most relevant recommendation in current user context. Since user and item data was already simplified, this step is relatively ease to perform when compared with making suggestions from the scratch.

Basic approach suggest to show random recommendations and with use of data mining methods derive ones which are most likely to be clicked. Unfortunately learning process would be extremely time-consuming in this case. Another approach is to narrow the set to only few possibilities (about which we expect high CTR) and to use learning techniques among them.

Let me introduce few behavioral hypothesis:

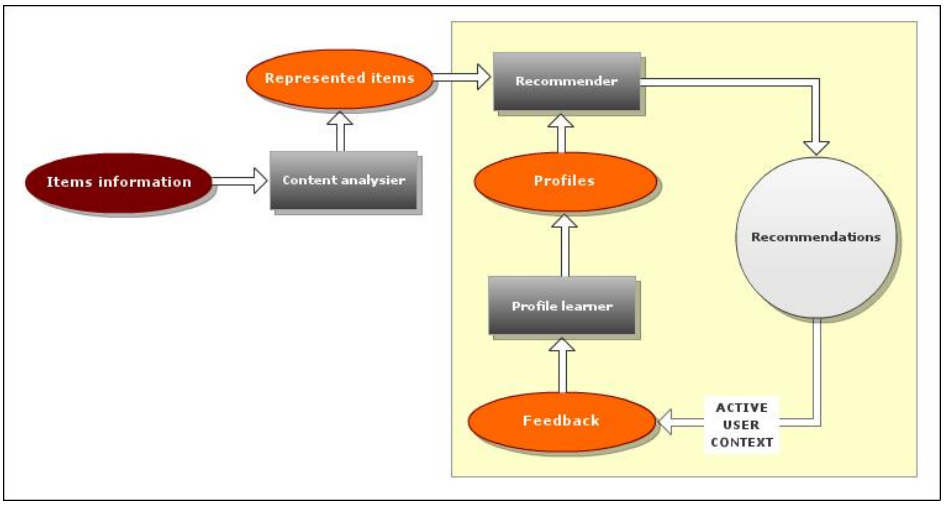

Figure 5.3: Recommender design

1. User is less likely to click in unknown link if he do not trust in the recommender,

2. User expects the recommender to show items he does not know,

3. and the items which he would like.

Additionally, most of users, who entered on venue details directly by search engine, were searching for exactly this entity. That is why it is rather hard to convince them to check something different. I suggest solution based on gaining trust first and showing the real suggestions afterwards.

Three strategies will be tested:

1. **current**, randomized one as described before (control group),

2. **most-popular**, when new visitor lands on business details site, at first we show him one set of most popular items as recommendations. Afterwards (if only he clicks on one of suggestions) we present to him real suggestion from category of the venue which he entered by,

3. **most-controversial**, similar to previous one, however instead of most popular item we show the one with highest variance of ratings.

We will estimate popularity with a number of reviews. The main goal is to attract the user with something that he already knows. Hopefully after reading opinions in accord with his taste, he will start to believe in the website's content. Since we redirect him to venues with high number of reviews, he probably will find there reviews to agree with. This scenario is even more likely to appear if the venue has big variance of ratings, that is why I decided to check both approaches.

In literature one can already find ideas which base on user's trust, as main factor of successful recommender. However, most of them focus on systems with more data available, varying trust for optimizing outcome from collaborative filtering. In this work I try to manipulate trust in the first step, where no information about the user is provided.

5.5. Results of the experiment

The experiment was conducted between 12th and 18th of September 2011 on the group of over 6000 new visitors. They were randomly assigned to one of three groups described in previous chapter "current", "most-popular" and "most-controversial".

group	exit rate	visitors	avg. time	CTR
current	71.64%	2008	00:00:21	0.30
most-popular	67.35%	2094	00:00:24	0.38
most-controversial	69.84%	2142	00:00:23	0.33

Table 5.1: Results of the experiment. Visitors column describes number of users in each group.

Hypothesis that substitution of recommendation algorithm does not change the exit rate was tested using Bernoulli test. I assume that probability of exit is equal to the randomized one (**current**), i.e. p = 0.7164. I tested two alternative hypothesis and using `binom.test` from `R` package obtained:

1. **most-popular** - $p\text{-}value = 0.00001483$,
2. **most-controversial** - $p\text{-}value = 0.04057$,

which implies significant difference in both cases. Analogical analysis shows importance of change basing on the CTR.

Now, applying new algorithm on the whole website, should decrease number of exits just after first visit by $1 - (67.35/71.64) = 5.98\%$ what might be considered as a satisfying result especially for such a relatively small change.

Chapter 6

Conclusions

In this work I surveyed the field of recommender systems putting focus on their mathematical and statistical background. I discussed whole process of choosing apropriate measure, algorithms suitable for given task and methodology of testing supported by real-life example. Additionally in chapter 4 several techniques were comared on the small scale dataset.

Two empirical studies presented in the end of this work, shown how one can obtain significant effects using recommender system. Information retrival and considerable analyse of website's structure and user's profile, may lead to astonishing results with small effort.

As a conclusion I would like to highlight importance of precise problem definition before applying any of recommendation methods. First thing to ask should be rather "Why and where does my user need me to recommend him something?", not "How to recommend him content most efectively?". Since our goal is to satisfy the user, the most important thing is to understand his needs and only then fulfill them.

It is also important to remember that recommender system is just helpful tool - not the way to make profit by itself. Content of good quality and proper presentation of products should be seen as it's inseparable part. For instance Google would mean nothing if there would be no valuable content to find over the Internet.

Due to many machine learning competitions, state-of-the-art algorithms performance is outstanding. However, the problem should not be percieved as a strictly algorithmic one. There is much to be done in the other science fields which intersect on recommendations, especially in psychology and sociology. Small additional information retrive from user's behaviour may have great impact on quality and effectivity of recommendations. In my opinion future reaserach should move towards more interdisciplinary approach.

Used algorithm

Two simple approaches to recommendations used on neib.org and sweetrs.org: neighbourhood-based (consisted of both user-based and item-based) and content-based. Both written in matlab.

```
function X = userbased( y,M,sizeX,swap,sim,neib)
% Neighbourhood based regression algorithm
%  calculates similarities between users and than rating for i as
%  r_u(i) = \sum_v sim(u,v) * rv(i) / sqrt(sim(u,u) * sim(v,v)

% Output
% X - matrix to be estimated
% Input
% y - sampled entries
% M - masking operator, applied to vectorized form of X
% sizeX - size of matrix to be reconstructed
% swap - switch to ITEM-BASED (default = 0)
% sim - similarity measure (0 - PC, 1 - WPC, 2 - cos sim, default = 0)
% neib - percentage of neibours taken into sum

    Y = reshape(M(y,2),sizeX);

    if nargin < 4
        swap = 0;
    end
    if nargin < 5
        sim = 0;
    end
    if nargin < 6
        neib = 0.8;
    end
    if swap
        Y = Y';
    end;

    if (sim < 2)
        corr = pearsonsim(Y,sim);
    else
        corr = cosinesim(Y);
    end;
```

```
    corr(corr < 0) = 0;

    rated = Y>0;
    usedcorr = corr * rated';
    X = corr * Y';
    X = X ./ usedcorr;
    X(usedcorr==0) = mean(y);
    X = X';

    if (swap)
        X = X';
    end;
end

function X = contentbased( y,M,sizeX,prop)
% Content based calssification algorithm
% argmax_i Y_i'PP
% where Y_i is matrix of ratings of all elements of Y equal to i and 0
% elsewhere

% Output
% X - matrix to be estimated
% Input
% y - sampled entries
% M - masking operator, applied to vectorized form of X
% sizeX - size of matrix to be reconstructed
% p - lp norm (default 1)
% prop - properties of items

    Y = reshape(M(y,2),sizeX);

    L = zeros(sizeX);
    X = zeros(sizeX);
    Z = Y';
    for i=1:5
        pref = (Z.*(Z == i)) * prop;
        LIKE = pref * prop';
        LIKE = LIKE';
        X(LIKE > L) = i;
        L(LIKE > L) = LIKE(LIKE > L);
    end

    X(not (X > 1)) = mean(y);
    X(X>5) = 5;
end
```

Bibliography

[1] Ricci F., Rokach L., Shapira B., Kantor P., *Recommender Systems Handbook*, Springer Science+Business Media, 2011

[2] Amatriain X., Jaimes A., Oliver N. and Pujol J., *Data Mining Methods for Recommender Systems*, Springer Science+Business Media, 2011

[3] Jolliffe I.T., *Principal Component Analysis*, Springer, 2nd ed., 2002

[4] Langseth H., *Bayesian Networks for Collaborative Filtering*, 2009

[5] Xia Z., Dong Y., Xing G., *Support vector machines for collaborative filtering*, ACM-SE 44 Proceedings of the 44th annual Southeast regional conference ACM New York, NY, USA 2006

[6] Ungar L.H., Foster D.P., *Clustering Methods for Collaborative Filtering*, AAAI Press, Menlo Park, California, 1998

[7] Mahmood, T., Ricci, F., *Improving recommender systems with adaptive conversational strategies*. In: C. Cattuto, G. Ruffo, F. Menczer (eds.) Hypertext, pp. 73–82. ACM (2009)

[8] McSherry, F., Mironov, I., *Differentially private recommender systems: building privacy into the net*. In: KDD '09: Proceedings of the *15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 627–636. ACM, New York, NY, USA (2009)

[9] Agarwal D., *Recommender Problems for Content Optimization*, http://www.stanford.edu/group/mmds/slides2010/Agarwal.pdf, 2010

[10] Vozalis M., Margaritis K., *A Recommender System using Principal Component Analysis*, http://www.stanford.edu/group/mmds/slides2010/Agarwal.pdf, 2007

[11] Berry M.W., Dumais S.T. & O'Brien G.W., *Using Linear Algebra for Intelligent Information Retrieval*, http://lsirwww.epfl.ch/courses/dis/2003ws/papers/ut-cs-94-270.pdf, 1994

[12] Brand, M., *Fast Online SVD Revisions for Lightweight Recommender Systems*, SIAM International Conference on Data Mining (SDM), May 2003

[13] Barbella D., Benzaid S., Christensen J., Jackson B., Qin V., Musicant D, *Understanding Support Vector Machine Classifications via a Recommender System-Like Approach*, http://www.cs.carleton.edu/faculty/dmusican/svmzen.pdf, May 2003

[14] Min Sung-Hwan, Han I., *Recommender Systems Using Support Vector Machine*, ICWE'2005. pp.387-393, 2005

[15] Cataldo M., *Enhanced Vector Space Models for Content-based Recommender Systems*, ACM RecSys'10, 2010

[16] Lops P., Gemmis M., Semeraro G., *Content-based Recommender Systems: State of the Art and Trends*, Recommender Systems Handbook p. 73, Springer Science+Business Media, LLC, 2011,

[17] Gofman A., *Consumer Driven Multivariate Landing Page Optimization: Overview, Issues, and Outlook*, International Journal of Technology Marketing, Volume 6, Number 1, August 2011, pp. 72-84(13)

[18] Richmond S., *YouTube users uploading two days of video every minute*, The Telegraph, http://www.telegraph.co.uk/technology/google/8536634/YouTube-users-uploading-two-days-of-video-every-minute.html, 2011

[19] Goldberg D., Nichols D., Oki B.M., Terry D., *Using collaborative filtering to weave an information tapestry*, Commun, ACM 35(12), 61–70, 1992

[20] Alpcan T., Xing Liu, *A Game Theoretic Recommendation System for Security Alert Dissemination*, Network and Service Security, N2S '09. International Conference, 2009.

[21] Torkaman A., Charkari N.M., Aghaeipour M., Hajati, E. *A recommender system for detection of leukemia based on cooperative game*, : Control and Automation, 2009. MED '09. 17th Mediterranean Conference, 2009.,

[22] Mukherjee R., Dutta P., Sen S., Sen I., *MOVIES2GO - A new approach to online movie recommendation*, In Proceedings of the IJCAI Workshop on Intelligent Techniques for Web Personalization, 2001.

[23] Harper M., Li X., Chen Y., Konstan J. *An Economic Model of User Rating in an Online Recommender System*, In Proceedings of The 10th International Conference on User Modeling, 2005

[24] Hernandez del Olmo F., Gaudios E., *Evaluation of recommender systems: A new approach*, Expert Systems with Applications 35 p. 790–804, 2008

[25] Montaner M., Lopez B., Lluis de la Rosa J., *Evaluation of recommender systems through simulated users*, http://eia.udg.es/ mmontane/montaner-iceis04.pdf, 2003

[26] Gunawardana A., Shani G., *Evaluating Recommendation Systems*, Journal of Machine Learning Research 10, 2009

[27] Das A., Mathieu C., Ricketts D., *Maximizing profit using recommender systems*, http://cs.brown.edu/people/aparna/rec.pdf, downloaded 29-07-2011

[28] Lehmann E.L., *Testing statistical hypothesis*, Ney York Wiley, 1970

[29] Kohavi R., *A study of cross-validation and bootstrap for accuracy estimation and model selection*, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence 2, 1995

[30] Nielsen J., *Usability Engineering*, Academic Press Inc, 1994

[31] Cox D.R., *Planning of experiments*, 1958

[32] Bailey R.A., *Design of Comparative Experiments*, Cambridge University Press, 2008

[33] Zhang J., Mueller S.T., *A note on ROC analysis and non-parametric estimate of sensitivity*. Psychometrika 70 (203-212), 2005

[34] Dellaroca C., *Strategic Manipulation of Internet Opinion Forums: Implications for Consumers and Firms*, MIT, 2004

[35] Pucci A., Gori M., Hagenbuchner M., Scarselli F., Tsoi A. C., *Applications of Graph Neural Networks to Large–Scale Recommender Systems Some Result*, http://www.wpi.edu/Pubs/ETD/Available/etd-0501102-130405/unrestricted/wkogel.pdf, downloaded 11-07-2011

[36] Kogel W., *Faster Training of Neural Networks for Recommender Systems*, Proceedings of the International Multiconference on Computer Science and Information Technology pp. 189–195, PIPS, 2006

[37] Fichtenholz G.M., *Differential und Integralrechnung*, Verlag Wissenschaft., 1964

[38] A. Frieze, R. Kannan, and S. Vempala., *Fast monte-carlo algorithms for finding low-rank approximations*, Journal of the ACM (JACM), 1998

[39] Sarwar, B., Karypis, G., Konstan, J., Reidl, J., *Item-based collaborative filtering recommenda-tion algorithms*, WWW '01: Proc. of the 10th Int. Conf. on World Wide Web, pp. 285–295

[40] Platt J., *Using Analytic QP and Sparseness to Speed Training of Support Vector Machines*, Advances in Neural Information Processing Systems 11, MIT Press, 1999

[41] Gurney, K., *An Introduction to Neural Networks*, London, Routledge, 1997

[42] Good, N., Schafer, J.B., Konstan, J.A., Borchers, A., Sarwar, B., Herlocker, J., Riedl, J., *Combining collaborative filtering with personal agents for better recommendations.* AAAI '99/IAAI '99: Proc. of the 16th National Conf. on Artificial Intelligence, pp. 439–446. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1999

[43] Bell R., Koren Y. and Volinsk C., *The BellKor solution to the Netflix Prize*, http://www2.research.att.com/ volinsky/netflix/ProgressPrize2007BellKorSolution.pdf, downloaded 1-09-2011

[44] Listens D., *The Napoleon Dynamite Problem*, http://blogs.oracle.com/plamere/entry/the_napoleon_dynamite_problem, downloaded 1-09-2011

[45] Bell R., Koren Y. and Volinsk C., *From the Labs: Winning the Netflix Prize*, http://goo.gl/svsIb, ATTTechChannel, downloaded 1-09-2011

[46] Koren, Y., *Factorization meets the neighborhood: a multifaceted collaborative filtering model.* KDD'08: Proceeding of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 426–434. ACM, New York, NY, USA, 2008

[47] Takacs, G., Pillaszy, I., Nemeth, B., Tikk, D., *Major components of the gravity recommendation system.* SIGKDD Exploration Newsletter 9(2), 80–83 (2007)

[48] Desrosiers C. and Karypis G., *A comprehensive survey of neighborhood-based recommendation methods*, http://glaros.dtc.umn.edu/gkhome/fetch/papers/NbrRSsurvey2011.pdf, downloaded 4-09-2011

[49] Billsus, D., Pazzani, M.J., *Learning collaborative information filters.* ICML '98: Proc. of the 15th Int. Conf. on Machine Learning, pp. 46–54. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998

[50] Netflix blog, *Netflix Prize Update*, http://blog.netflix.com/2010/03/this-is-neil-hunt-chief-product-officer.html, downloaded 07-09-2011

[51] YehudaKoren, *How useful is a lower RMSE?*, http://www.netflixprize.com/community/viewtopic.php?id=828, downloaded 07-09-2011

[52] Lang, K., *News Weeder: Learning to filter netnews.* In: Proc. of the 12th Int. Conf. on Machine Learning, pp. 331–339. Morgan Kaufmann publishers Inc., San Mateo, CA, USA, 1995

[53] Majumdar A., *Matrix Completion via Thresholding*, http://www.mathworks.com/matlabcentral/fileexchange/26395-matrix-completion-via-thresholding, downloaded 2-09-2011