Recommendation Systems: Prediction of Web Site User Preferences

Sebastian Ernst[1], **Dominik Pacewicz**[2], **Radosław Klimek**[3]

[1] Computer Science PhD student, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, AGH-UST, Al. Mickiewicza 30, 30-059 Kraków, sebi@selex.pl
[2] domp@selex.pl
[3] Laboratory of Computer Science, Department of Automatics, AGH-UST, Al. Mickiewicza 30, 30-059 Kraków, rklimek@agh.edu.pl

Abstract. *The rapid development of information propagation technologies, especially the Internet has produced an overwhelming amount of data and information. An average user has a hard time to choose only the interesting and relevant items.*

Retrieval of specific data is usually made possible by submitting queries to search engines. A problem appears when a collection of data contains similar items which cannot be evaluated using a uniform rating scale. It is especially apparent in online shops offering books, music records or films. Such items are rated on a purely subjective basis – it is impossible to provide a rating scale that's universal, but at the same time, takes the preferences of individuals into account.

A recommendation system is a piece of software used to predict the attractiveness of specific items as perceived by specific users. Input data is sparse and contains user ratings of items. Recommendation systems use, among other methods, collaborative filtering. This method locates users with similar preferences, based on already collected item ratings. They are used to predict the missing values in the customer/product matrices. This enables the users to find items they otherwise wouldn't be able to locate.

There are two main problems related to implementation of collaborative filtering recommendation systems. The first one is data sparsity, which greatly reduces relevance of the results, especially in initial stages of operation. The second one is computational complexity of methods used.

This article shows and compares different approaches to implementation of recommendation systems, discusses possible problems and describes the mathematical, statistical and computing methods of solving them. It also shows other applications of collaborative filtering, such as remote education and web search engines and shows directions of improvement of CF-based recommendation systems.

1. Introduction The beginning of 1990s saw the dawning of the age of the World Wide Web. It was – and still is – hard to find anyone who wouldn't be impressed by the availability of search engines and vast collections of literally all kinds of data. However, it wasn't long before the amount of data could, and did, easily overwhelm an average user of the Internet.

Traditional search methods rely on simple content classification routines. For websites, that would be keyword or full-text search. Online shops, especially those dealing with books, films or music, tend to rely on a uniform rating scale. The accuracy of web search is largely dependent on proper formulation of search queries, which can be rather difficult. The real problem appears where the relevance of results depends on the taste of the user, and therefore ratings are different for every single person.

Of course, it is possible to derive a proprietary classification scheme to group similar items – e.g. movie or music genres. That, however, doesn't solve all the problems, firstly – because most people can't tell what exactly their favourite genres are, and secondly – because in such scheme, every item would have to be first evaluated and classified by a group of experts.

This is recommendation systems kick in. The underlying idea is quite simple: we want a piece of software that will collect item ratings from

users and apply them to other items in order to match the taste of a particular person. This approach enables automatic classification of items, usually making extra user input unnecessary. Such a system uses previously collected data to classify and recommend new items to users, so data sparsity can be a problem, especially in the initial stage of operation.

Recommendation systems are often used in online applications, so special techniques have to be used in order to minimize computational complexity and maintain a short response time.

The following article presents several approaches, methods and applications used to achieve this goal.

2. Collaborative filtering One of the methods most often used to derive user-specific recommendations from a subset of data is collaborative filtering, first introduced in the mid eighties [1]. The basic concept is similarity between users and rated items. For instance, let's say a user likes movies A and B. We can assume that other users who also liked those movies have a similar taste to the user in question. On that basis, it should be possible to provide the user with other movie recommendations. [8]

2.1. Data representation In a typical recommendation system, consisting of m users and n items, data is represented as a $m \times n$ user/item matrix, R. In the simplest case, the matrix is binary: element r_{ij}, equals 1 if the user was interested in a product or 0 otherwise. Usually, this amounts to the fact of a customer purchasing a given product. [6]

Some systems extend this idea by indicating not only the fact of a user being interested in an item, but also how satisfied they were with the choice. This is done by introducing a rating scale – either a discrete or, as proposed by the authors of [9], a continuous one. In this case, before further processing, the ratings need to be normalized by subtracting their mean ratings among all users and dividing them by their standard deviation.

2.2. Initial processing One of the problems of collaborative filtering recommendation systems is the fact that the ratings data is usually sparse. In large systems, ratings from a single users encompass less than 1 percent of all available items – usually less. This can make it difficult or even impossible to provide ratings for a particular user.

Another problem related to data sparsity is the lack of transitivity. Let's say user A correlates with user B, and user B correlates with user C. With a sparse dataset, we can't assume user A will correlate with user C, as they may not have enough common rated items, which will additionally limit the capabilities of the recommendation algorithm.

2.2.1. The 'gauge set' approach One approach presented in [9], is to provide a uniform set of items, rated – usually upon registration – by all users. This set is called a *gauge set* and has to meet several prerequisites. First of all, users will rate the gauge set items based solely on their descriptions, as they are assumed not to have had personal experience with any of them. Therefore, the descriptions have to be unbiased, preferably short and as easy to understand as possible.

The authors claim that this method proved to yield satisfactory results in their model implementation – a joke recommending system, Jester. There are, however, a few possible problems. Firstly, the efficiency of the initial user profiling procedure depends on the selection of items forming the gauge set. Ideally, their characteristics should reflect the characteristics of all items. However, in quickly-developing systems (online shops, e-learning repositories), the gauge set could prove irrelevant after a while, but it couldn't be changed, as only that would result in inconsistent ratings for the 'old' and 'new' users.

Another thing worth considering is the feasibility of actually getting users to rate the items in the gauge set. This could be a problem in corporate or business-to-business services, as such approach could seem plainly unprofessional, as well as in widely-available online systems, as users are often discouraged by having to go through a long registration/profiling procedure. Therefore, in some cases it is better to use only ratings

provided by users on a normal basis, i.e. in the course of system operation, even sacrificing effectiveness in the initial phase.

2.2.2. Computation of similarity Similarity between users is usually determined by computing a Pearson correlation matrix of commonly rated items (or items which form the gauge set). Each element of the matrix is the Pearson product-moment correlation coefficient between a user and an item.

If A is the normalized subset of R consisting either of commonly rated items or the items which form the gauge set, the correlation matrix C can be defined as [9]:

$$C = \frac{A^T A}{n-1}$$

Another approach to measuring user similarity is to consider two users (i and j) as two vectors in a n-dimensional space. In this case, the angle will be narrow between similar users' vectors and wide between those of users whose preferences differ.

$$C_{ij} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{||\vec{i}||_2 * ||\vec{j}||_2}$$

Both methods result in a symmetric square matrix, which defines the similarity between two random variables – in our case, preferences of two users.

2.2.3. Principal component analysis Although a computed global similarity matrix well represents similarities of users' preferences, it is quite difficult to analyse, as every single recommendation would require a lot of additional lookups and calculations. Therefore, similarity matrices are often subjected to dimension-reducing transformations, among which the most popular is principal component analysis (PCA).

Introduced in the beginning of the twentieth century by Karl Pearson [2], it was later generalized by Harold Hotelling [3], thus the common alternative name – the Hotelling transform.

PCA is the optimal linear transformation for reducing multi-dimensional data into a simpler coordinate system, while keeping the subspace with the greatest variance. In practice, calculation of components is performed by finding the eigenvalues and eigenvectors of the covariance matrix, described in 2.2.3. Eigenvectors with the largest eigenvalues correspond to the most correlated dimensions of the source data. In almost all cases, a small number of eigenvectors constitute a large part of the total variance.

Then, original data is projected onto a vector space with a reduced number of dimensions. Usually, data is projected onto a two-dimensional plane, as it enables easy clustering of data (e.g. users) and makes it possible to create a plot for visual analysis and representation.

PCA and eigenvector computation is used by the method described in [9], hence the name.

2.3. Neighbourhood formation In order to provide a specific user with personalised recommendations, we need to find other users with preferences as similar as possible. In other words, we want to form a neighbourhood for a given user, e.g. create a list of other like-minded users, sorted according to similarity. Two most commonly used types are central and aggregate neighbourhoods. A central neighbourhood simply consists of a given number of users with the highest similarity (computed as described in 2.2.3). An aggregate neighbourhood, consisting of l elements, is constructed using the following routine. In the first iteration, we select the user that's closest to the starting user. Then, in every subsequent iteration, the algorithm computes the centroid of the current neighbourhood and selects the user that's nearest to the centroid as the next member, until the neighbourhood is of desired size. [6]

3. Scalability and clustering Most online systems that could profit from implementation of collaborative filtering routines develop very quickly. For instance, an online shop isn't unlikely to exceed a million of items and users in a relatively short time.

The response time for every web-based systems should be as short as possible. Computational complexity of nearest neighbour algo-

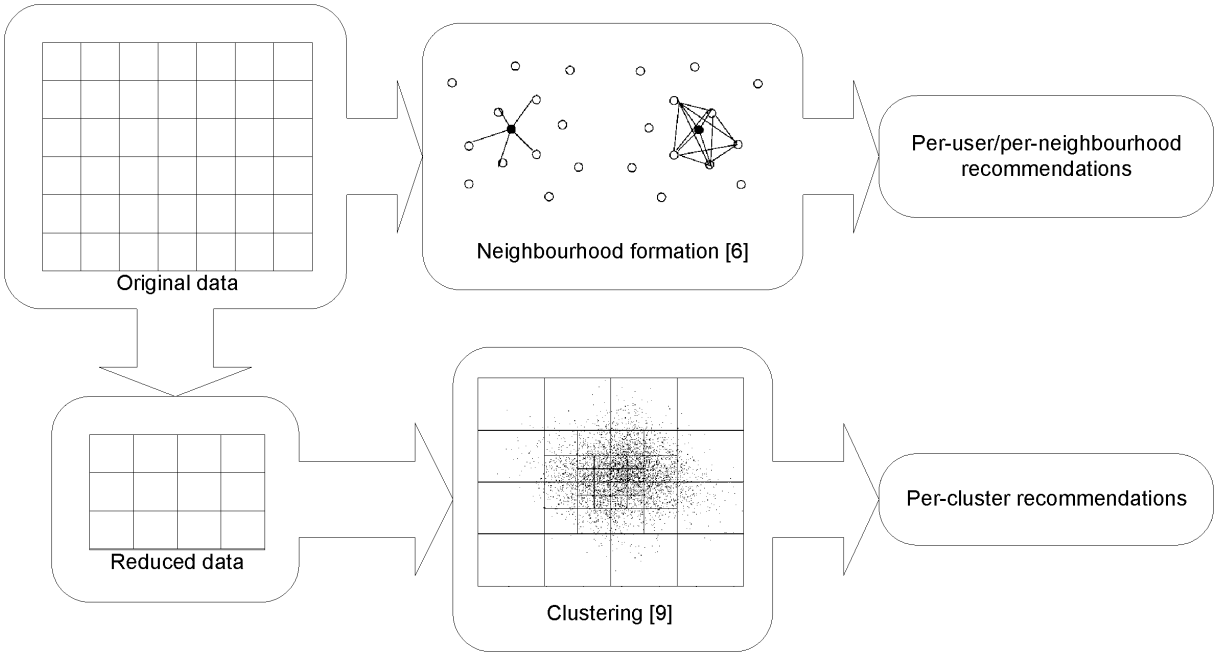

Fig. 1. Basic operation stages of CF-based recommendation systems

rithms grows with both the number of items and the number of users. [6]

3.1. Clustering In order to decrease the complexity of computations, users in the system can be arranged into clusters, according to similar preferences. This is easiest to achieve after reducing the number of dimensions, e.g. using principal component analysis (2.2.3).

There are several clustering methods available. Two-dimensional data, as presented in [9], can be easily grouped into clusters using a recursive algorithm implementing rectangular clusters. The size of the cells decreases near the centre (origin) of the plane.

The algorithm is simple. First, the rectangle enclosing all the points is bisected both vertically and horizontally, in order to create four sub-cells. Then, in every iteration, each of the four cells which have the origin as one of the vertices is subdivided into four new cells, until the desired number of clusters is achieved.

There are other clustering methods for collaborative filtering, as described in [5]. The authors have established that the accuracy can be improved using Gibbs sampling, but the current algorithms are very computionally expensive. More efficient Gibbs sampling methods are being developed.

3.2. Offline computation As shown above, quick response, which is critical for online applications, is hard to achieve if all calculations have to be done in real time. That's why recommendation systems are often split into two modules.

One, which is periodically run offline, performs all the necessary calculations, including similarity computation, dimension reduction and assignment of users into clusters. This process can be run with a lower priority and utilize the idle processor time of the web server.

The second module operates online. For users who have previously been using the system, all that needs to done is retrieve the recommendations computed using the offline module and present them to the user. Therefore, the computational complexity is $O(1)$.

Obviously, new users entering the system cannot be provided with relevant recommendations, as the system knows nothing about their preferences.

For algorithms using the 'gauge set' approach (see 2.2.1), the procedure for new users looks as follows [9]:

1. Ratings are collected for all items in the gauge set,
2. PCA (2.2.3) is used to project the vector onto the plane,
3. User is assigned to the appropriate cluster.

4. Applications and existing implementations Collaborative filtering has a virtually endless list of applications. However, there are some areas where collaborative filtering recommendation systems are particularly useful.

4.1. E-commerce and taste-based evaluation The most obvious application, which also served the purpose of the example throughout this article, is e-commerce. Online shop usually offer items which are subject to taste – this is most clearly visible in case of books, music recordings, movies, etc.

Collaborative filtering is used by most large online shops. Some well-known examples incldue Amazon, Barnes and Noble, Netflix and Hollywood Video. CF is also becoming increasingly popular in non-commercial purposes. Audioscrobbler is a service used solely for the purpose of recommending music to users, based on their preferences. Profiles are created automatically, based on what the users listen to on their computers. It also provides means of communications for users with similar tastes.

4.2. E-learning Electronic training and education programmes are becoming increasingly popular. Due to their nature, they are largely based on various repositories of learning objects. A learning object is any element that can be contained within an e-learning course: a video, an image, a map or a website. [4, 8]

In case of e-learning, recommendations are computed based on the topic, as well as on preferences of a given user. E-learning developers can therefore create courses which are best suited to a given customer/student, and therefore improve the efficiency of education.

4.3. Web search The World Wide Web continues to grow at an overwhelming pace. If it wasn't for efficient search engines, such as Google, most users would have a hard time trying to find any useful information at all. Unfortunately, web search engines, being purely text-based, often provide many results which seem relevant, but really are useless.

There are two problems related to implementation of collaborative filtering in web search engine. Firstly, different users use different search queries to locate the same information. The problem of synonymy has been pointed out in [6]. Therefore, web search engines would have to perform additional filtering of search queries, based on known synonyms.

The other problem is that there are no means of monitoring user sessions among multiple servers. All statistics a search engine can collect are "user clicks" – which results are usually chosen by the user for a given query. There are some attempts to provide a client-side platform for collecting automatic feedback from users. This is the case with Google Toolbar, which monitors users' behaviour in regard to search queries and sends in anonymous statistics back to the server.

5. Evaluation methods Evaluation of recommendation systems is a difficult task. Almost all recommendation systems perform differently for different data sets. One of the main characteristics of a data set is the number of users in relation to the number of items.

5.1. System expectations Additionally, recommendation systems created for different purposes have to meet different requirements. [7] provides a simple classification of recommendation systems, according to their purpose. Usually, a recommendation system is expected to rule out as many irrelevant results as possible, even by sacrificing some which are good. This approach is described as *Find Good Items*. It also provides an example of a lawyer looking for precedents. In this case, the goal is not to overlook a single case which could prove useful – a *Find All Good Items* approach.

Another goal is to recommend a sequence of items that's attractive as a whole, rather than focusing on individual, but not well-matched recommendations. This is often used by personal-

ized Internet radio stations. Authors of [7] aren't aware of any systems built specifically for this task.

Recommender credibility is often a very important issue for users. Users often 'play' with systems, in order to evaluate the algorithms use and system responsiveness. Some systems are optimized to recommend items the user doesn't yet know about – as useful as it may be, this feature could cause the results to be perceived by some users as inaccurate.

5.2. Accuracy metrics There are several formal accuracy metrics which can be used to evaluate recommendation systems. One of the most commonly used is Mean Absolute Error (MAE), which measures absolute deviation between the predicted value and the actual rating.

This, as described in [8], is usually performed by hiding one of the ratings from the dataset and compare the prediction results with the actual value.

Classification Accuracy Metrics measure the frequency of correct recommendations and are appropriate for systems with binary user preferences. Such metrics don't measure the accuracy of predictions itself – just the classification of items. They can, however be challenged by data sparsity.

Other metrics include precision and recall related measures, ROC curves and ad-hoc classification accuracy measures, as described in [herlocker].

6. Conclusion Recommendation systems have a practically endless list of applications. Soon, it will be impossible to create a good e-commerce tool or a search engine without implementing collaboration filtering routines.

Two aspects of recommendation systems that still need improvement are scalability and methods of dealing with sparse datasets. The answer to both these problems is reduction of mutidimensional data to a lower number of dimensions and grouping of items and users into clusters. Data should be reduced as early after collection as possible, while maintaining a low rate of data loss.

There also is space for improvement in initial user profiling. Systems which don't incorporate such procedures are unable to provide relevant suggestions before the user has, for instance, purchased at least several items. However, initial profiling procedures (e.g. using the 'gauge set' approach) should be as clear and not time-consuming as possible. We are seeking methods to develop efficient user profiling procedures that could be incorporated into to the user registration procedure. Surveys have shown that users are likely to accept simple questions with two alternative answers. In order to automate this process, we are looking for ways of preparing such questions based on already collected user evaluation data.

References.

[1] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, *Grouplens: An open architecture for collaborative filtering of netnews,* (Proceedings of the 1994 ACM conference on Computer supported cooperative work).

[2] K. Pearson, *On lines and planes of closest fit to systems of points in space* (The Philosophical Magazine, Series 6, Volume 2, Number 11, 1901).

[3] H. Hotelling, *Analysis of a complex of statistical variables into principal components* (Journal of Educational Psychology 24, 1933).

[4] S. Downes, *Learning objects: Resources for distance education worldwide* (International Review of Research in Open and Distance Learning, 2001).

[5] L. Ungar, D. Foster, *Clustering methods for collaborative filtering* (Proceedings of the Workshop on Recommendation Systems. AAAI Press, 1998).

[6] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, *Analysis of recommendation algorithms for e-commerce* (ACM Conference on Electronic Commerce, 2000).

[7] J. Herlocker, J. Konstan, L. Terveen, J. Riedl, *Evaluating collaborative filtering recommender systems* (ACM Transactions on Information Systems, 2004).

[8] M. Anderson, M. Ball, H. Boley, S. Greene, N. Howse, S. McGrath, D. Lemire, *RACOFI: A Rule-Applying Collaborative Filtering System* (International Workshop on Collaboration Agents: Autonomous Agents for Collaborative Environments, 2003).

[9] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, *Eigentaste: A constant time collaborative filtering algorithm* (Information Retrieval, 2001).