Proceedings of the International Multiconference on Computer Science and Information Technology pp. 189–195
ISSN 1896-7094

Applications of Graph Neural Networks to Large–Scale Recommender Systems Some Results

A. Pucci[1], M. Gori[1], M. Hagenbuchner[2],
F. Scarselli[1], and A. C. Tsoi[3]

[1] Dipartimento di Ingegneria dell'Informazione, University of Siena,
Via Roma, 56. Siena, Italy
{augusto,marco,franco}@dii.unisi.it
[2] Faculty of Informatics, University of Wollongong,
Wollongong NSW 2522, Australia
markus@uow.edu.au
[3] e-Research Centre, Monash University
Victoria 3800, Australia
ahchung.tsoi@adm.monash.edu.au

Abstract. Recent machine learning algorithms exploit relational information within a graph based data model. This paper considers one of the most promising supervised machine learning methods, the *graph neural networks* (GNN), for this class of problems. An application of the GNN approach to a recommender system problem revealed certain limitations of the GNN. A recommender system makes personalized product suggestions by extracting knowledge from previous user interactions. This is a specially interesting scenario because of the scale-free nature of graphs modelling user preference dependencies. We focused on the MovieLens dataset, which contains data collected from a popular recommender system on movies, and has been widely used as a benchmark problem for evaluating recently proposed approaches. This paper performs a deep analysis on the dataset to help discovery of some intriguing properties, and discusses problems and limitations encountered by GNN while facing this particular practical problem.

1 Introduction

Recent developments in machine learning approaches produced models capable of exploiting relational information within a graph based data model. This is an issue of particular interest since for many practical learning problems it is not possible to obtain a sufficient number of examples to train a model. The idea is to balance a limited number of examples by providing relationship information among the data. One of the most promising algorithms belonging to this class is the Graph Neural Network (GNN). In this paper we apply the GNN algorithm to a particular class of applications: recommender systems. A recommender system (RS) makes personalized product suggestions by extracting knowledge from previous users interactions. Such services are particularly useful in the modern electronic marketplace which offers an unprecedented range of products. Several recommender systems have been developed that cope with different products, e.g. MovieLens for Movies (see [7]), GroupLens for usenet news [10], Ringo for music [8], Jester for jokes [11] and many others (see [9] for a recent review). Formally, a RS deals with a set of users u_i, $i = 1, \ldots, U_n$ and a set of products p_j, $j = 1, \ldots, P_n$, and its goal consists of computing, for each pair i, j, a rating $\hat{r}_{i,j}$ that measures the expected interest of users u_i for product p_j. A RS constructs a user profile on the basis of explicit or implicit interactions of the user with the system. The profile is used to find products to recommend to the user. The most successful and well–known approach to RS design is based on the collaborative filtering approach [7, 8]. In collaborative filtering, each user collaborates with others to establish the quality of products by providing his/her opinion on a set of products. Recommender system is a specially interesting scenario because of the scale-free nature of graphs modelling user preference dependencies. Despite this problem there are spectral graph based approaches which obtain good performances [5]. This paper shows how this graph topology can negatively affect the learning capability of a GNN when applied to a RS. We will use the MovieLens dataset for this evaluation; this data set has been widely used as a benchmark for evaluating recently proposed approaches to the RS problem.

This paper is organized as follows: Section 2 briefly introduces some ideas about GNNs, Section 3 analyses the MovieLens dataset, Section 4 describes a way to model the learning problem for use with GNNs, then Section 5 discusses some experimental results. Finally, Section 6 draws some conclusions.

2 Graph Neural Networks

Graph Neural Networks (GNN) are a class of recently proposed connectionist models that can directly process a set of patterns along with their relationships [6]. GNN extends recursive neural networks and can be applied on most practically useful kinds of graphs, including directed, undirected, labelled and cyclic graphs. In the following, $|\cdot|$ represents the module or the cardinality operator according to whether it is applied on a real number or a set, respectively. The one-norm of vector v is denoted by $\|v\|_1$, i.e. $\|v\|_1 = \sum_i |v_i|$. Let G be a directed graph[4]. Connections to a given node n of G are represented by $ne[n]$. Each node n may have a label that is denoted by $l_n \in R^q$. Usually labels include features of the object corresponding to the node.

The main idea underlying GNNs is to attach to each node v a vector $x_v \in R^s$, called *state*, which collects a representation of the object denoted by v. In order to define x_v, we assume that the arcs denote causal dependence relationships between the objects represented by nodes. Thus, x_v is naturally specified using the information contained in the label of v and in the parent nodes of v. More precisely, let w be a set of parameters and f_w be a parametric *transition function* that expresses the dependence of a node on its neighborhood. The state x_v is defined by the solution of the system of equations:

$$x_v = f_w(l_v, x_{pa[v]}, l_{pa[v]}),\ v \in V \tag{1}$$

where V is the set of nodes of G, and l_v, $x_{pa[v]}$, $l_{pa[n]}$ are the labels of v, the states and the nodes having an arc directed towards v respectively. For each node v, an output vector $y_v \in R^m$ is also defined which depends on the state x_v and the label l_v. The dependence is described by a parametric *output function* g_w

$$y_n = g_w(x_v, l_v),\ v \in V. \tag{2}$$

Let x, l and y be the vectors constructed by stacking all the states, all the labels and all the outputs, respectively. Then, Equations (1) and (2) can be written as:

$$\begin{aligned} x &= F_w(x, l) \\ y &= G_w(x, l) \end{aligned} \tag{3}$$

where F_w and G_w are the composition of $|V|$ instances of f_w and g_w, respectively. Notice that x is correctly defined only if the solution of system (3) is unique. The key choice adopted in the GNN approach consists of designing f_w such that F_w is a contraction mapping[5] with respect to the state x. In fact, the Banach fixed point theorem guarantees that if F_w is a contraction mapping, then Eq. (3) has a solution and the solution is unique. Thus, Eqs. (1) and (2) define a method to produce an output y_n for each node, i.e. they realize a parametric function $\varphi_w(G, n) = y_n$ which operates on graphs. GNN model uses a Jacobi algorithm [12] for sparse nonlinear systems to compute the state x and the output y and a specialized version of the Almeida–Pineda algorithm [3, 4] is used to learn the parameters. More details can be found in [6]. Here, it is worth mentioning that whereas the learning procedure may be time consuming, the computation of $\varphi_w(G, n)$ is a relatively fast process. In fact, Jacobi algorithm is particularly efficient and is successfully applied even on sparse matrices with billions of variables [13]. Finally, GNNs have been proven to be universal approximators and are able, under mild assumptions, to approximate in probability any function on graphs [6].

3 MovieLens Data Set

The MovieLens dataset is a popular benchmark problem on which many recently proposed approaches to RS were evaluated[6] (e.g. [5, 1]). The schema of the MovieLens dataset resembles the structure of many other collaborative filtering applications. Hence our results apply to environments having the same data structure as MovieLens. The dataset has been constructed by considering only users who rated 20 or more movies, in order to achieve a greater reliability on user profiling. The dataset contains over $100,000$ ratings from 943 users for $1,682$ movies. Every opinion is represented by a tuple: $(u_i, m_j, r_{i,j})$, where u_i is the i-the user, m_j is the j-the movie, and $r_{i,j}$ is an integer score ranging between 1 (bad movie) and 5

[4] The general version of GNNs can process both directed and undirected graphs. Here, we assume that arcs are directed to simplify the presentation.

[5] A function $l : R^a \to R^a$ is a contraction mapping with respect to a vector norm $\|\cdot\|$, if there is a real μ, $0 \le \mu < 1$, such that for any $y_1 \in R^a$, $y_2 \in R^a$, $\|l(y_1) - l(y_2)\| \le \mu\|y_1 - y_2\|$.

[6] The dataset is available from `http://www.movielens.umn.edu`

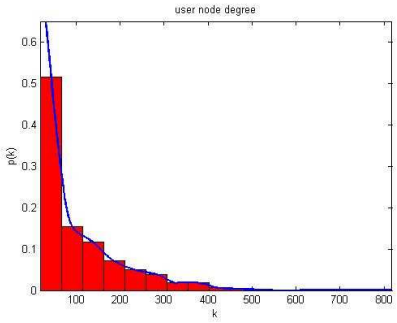


Fig. 1. Probability distribution of the number of movies rated by a user.

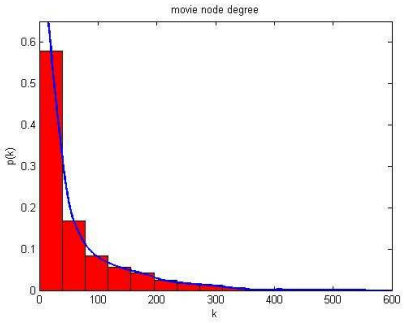


Fig. 2. Probability distribution of the number of ratings received by a movie.

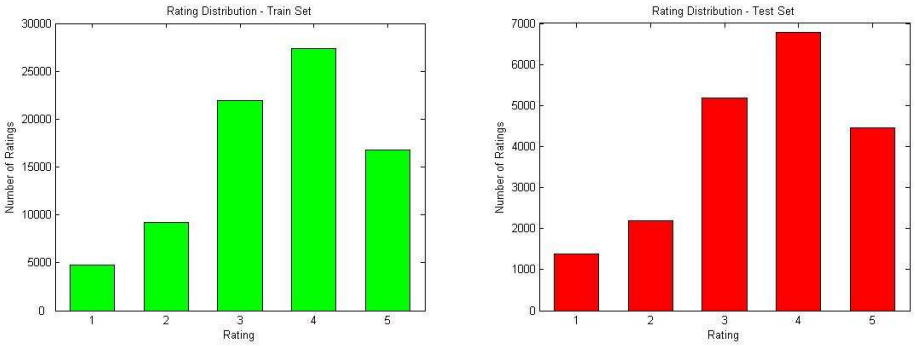


Fig. 3. Number of ratings for each score in the training data set (left) and the testing data set (right).

(good movie) concerning the j-the movie by user i. The database provides a set of features characterizing users and movies which include: the category of the movie, the age, gender, occupation, and postcode of the user. The dataset comes with five predefined splits, each uses 80% of the ratings for the training set (Tr) and 20% for the test set (Te). A statistical analysis has been carried out to understand some of the properties of this dataset. Figure 1 and Figure 2 show respectively the probability distribution of the number of movies rated by a user and the probability distribution of the number of ratings received by a movie. It is worth noting that the distributions clearly follow a power law[7]. Networks where the node degrees show a power law distribution are often encountered in social networks [2]. For example, the number of hyperlinks referencing a web page and the number of hyperlinks included in web pages have this property. The presence of a power law suggests that the network may evolve according to a preferential attachment mechanism[8]. In fact, it is likely that the number of ratings received by a movie increases proportionally to its popularity, i.e. the ratings already received; similarly, the number of ratings given by a user increases proportionally to its previous experience with the system, i.e. the current number of opinions the user has provided. Moreover, it is worth noting that most of the considered ratings (both in training and testing datasets) involve movies and users which are highly connected to other nodes in the graph G (that is more than 50 connections). In fact if we consider the whole 100,000 ratings in the data set (the training dataset together with the testing dataset), the average number of scores per movie is 60 and the average number of scores per user is 106. Figure 3 displays the distribution of the ratings in a training data set and in the corresponding testing data set, respectively. The histogram presents the number of ratings with the admissible scores $1, 2, 3, 4, 5$. It is observed that most of the ratings correspond to the scores 3 and 4, while 1 is the least selected rate. Such a trend is confirmed by the average of the ratings over the entire archive being 3.53. It is worth noting that the ratings are quite biased and their distributions are not uniform, which is an important consideration in view of the machine learning approach used in this paper.

[7] The relationship between two variables $x, y \in R$ follow a power law if it can be written as $y = x^b$, where $b \in R$.

[8] A network evolves according to a preferential attachment rule if the probability of introducing a new arc (u, v) is proportional to degree of u. Networks growing according to a preferential attachment rule show a power law distribution.

Table 1. Feature set used in the GNNs

usrDegree	The number of movies rated by a user
movDegree	The number of ratings received by a movie
usrMean	The medium score assigned by a user.
movMean	The medium score received by a movie
movCat	The category (or categories) the movie belongs to.
usrMean4Category	The average score assigned by a user to each movie category
usrSex	The user gender
usrJob	The user occupation
usrAge	The user age

Table 2. Feature set variants used in the GNNs

GNNNoData=[nodeType]
GNNOnlyData=[nodeType,usrSex,usrAge,usrJob,movCat]
GNNDegree=[nodeType,usrDegree,movDegree]
GNNMean=[nodeType,usrMean,movMean]
GNNMean4Category=[nodeType,usrMean,usrMean4Category,movMean,movCat]
GNNMeanAndData=[nodeType,usrMean,usrSex,usrAge,usrJob,movMean,movCat]

4 Data Model

We model the MovieLens data set by extracting both numerical and relational information. Relational information among data can be modeled by building a movie/user graph G. Two graphs are obtained: G_{Tr} and G_{Te} for the training and testing datasets respectively, where $G_{Tr} \subset G_{Te}$. G_{Tr} is built as follows: the graph contains a set of user nodes u_i, $i = 1, \ldots, U_n$ corresponding to users, and a set of movie nodes m_j, $j = 1, \ldots, M_n$ corresponding to movies in the system. Moreover we introduce rating nodes $s_{i,j}$, where every node $s_{i,j}$ is associated with a rating value $r_{i,j}$ [9]. So we have one rating node for every opinion expressed by a user u_i about a movie m_j in the training dataset Tr, i.e. the tuple $(u_i, m_j, r_{i,j})$. From the edges' point of view, for every tuple $(u_i, m_j, r_{i,j}) \in Tr$ we have four edges[10]: (u_i, m_j), (m_j, u_i), $(u_i, s_{i,j})$ and $(m_j, s_{i,j})$. Now it is very easy to define graph Te as an extension of graph Tr; in fact Te contains every node and edge that is already present in Tr, moreover for every $(u_i, m_j, r_{i,j}) \in Te$ we add a node $s_{i,j}$ (and also u_i and m_j if not already present) and only two edges (u_i, m_j) and (m_j, u_i). G_{Tr} and G_{Te} are the results of this building process. Computing the total number of edges and nodes is simple: if, for simplicity, we consider the case where every movie and user is present in at least one tuple in Tr, then we obtain 943 user nodes in G_{Tr} and G_{Te}, 1,682 movie nodes in G_{Tr} and G_{Te}, 80,000 rating nodes in G_{Tr} and 100,000 in G_{Te}, 320,000 edges in G_{Tr}, and 520,000 in G_{Te}.

As described in Section 2, a GNN implements a real parametric function $\varphi_w(G, v) \in R$, where G is a graph and v is one of its nodes. The goal of the corresponding machine learning problem consists of adapting the parameters w such that φ_w approximates some node targets. So in our case, the targets are defined only for the nodes $s_{i,j}$ denoting the opinions: it is required $\varphi_w(G, s_{i,j}) = \hat{r}_{i,j} \approx r_{i,j}$, where $r_{i,j}$ is the score corresponding to opinion $s_{i,j}$. The GNN is trained by using graph G_{Tr} and evaluated using graph G_{Te}. Moreover a GNN can also consider nodes with labels that define the properties of the corresponding objects. So we can also take into account the following features related to movies and users as reported in Table 1. In the experiments, we tested the variants reported in Table 2.

5 Experimental Results

For the experimentation, we measured GNNs' performance on the recommender system task described in Section 3. Our primary aim is to verify if a GNN will outperform a corresponding feed-forward neural network (FNN) on the same task. A simple FNN will not take into account the structural information; it

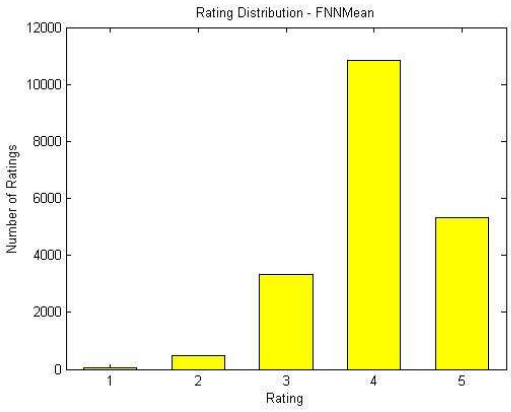
[9] Note the difference between $s_{i,j}$, that refers to a node in the graph and $r_{i,j}$ that is the associated score value.

[10] Here the direction of the arcs specifies the dependence of an object on another. Thus, G does not contain the inverse arcs $(s_{i,j}, u_i)$, $(s_{i,j}, m_j)$, since we assume that users and movies define their opinions, but opinions do not affect movies and users.

Table 3. The results by the FNN models and the GNN models compared with the other existing algorithms. Smaller MAE corresponds to a better performance.

Algorithm	MAE
CF [1]	0.7455
GNNMean	0.7555
FNNMean	0.7586
GNNMeanAndData	0.7597
FNNMeanAndData	0.7609
GNNMean4Category	0.8153
FNNMean4Category	0.8175
GNNDegree	0.8883
FNNDegree	0.8906
GNNOnlyData	0.9121
FNNOnlyData	0.9139
GNNNoData	1.4418

Table 4. The average number of ratings with score $1, 2, 3, 4$ and 5 obtained by the FNN-Mean model.



will only consider a set of vectors, one for every tuple in Tr. So we define FNNMean, FNNMeanAndData, FNNMean4Category, FNNDegree and FNNOnlyData in vectorial form by analogy with the corresponding GNN version.

The models based on FNNs used a two layered (one hidden layer) feedforward neural network with 10 hidden neurons, hyperbolic tangent activation function in the hidden layer and a linear activation function in the output node. Similarly, for the GNNs the transition and the output functions are implemented by two layered feedforward neural networks with 10 hidden neurons. These network architectures were selected by a trial and error procedure where it was found that larger architectures do not substantially increase the GNN performances. The results were evaluated using the *Mean Absolute Error* (MAE), which is defined as follows:

$$MAE = \frac{1}{\#L} \sum_{(i,j) \in L} |r_{i,j} - \hat{r}_{i,j})|$$

where L is the set of indices of the opinions in the testing data set and $\#L$ is its cardinality, $r_{i,j}$ is the score assigned to movie m_j by user u_i and $\hat{r}_{i,j}$ is the estimated value for $r_{i,j}$. MAE was preferred to other error measures [8], since it is an intuitive metric, and is the most popular measure of the quality for recommender systems [7]. So following [1], a 10-fold cross validation was conducted by randomly choosing each time different training and test data sets from the original standard splits and taking the average of the MAE. Table 3 reports the MAE averaged over the 10 trials for several models: we do not provide variance values since these are between 10^{-6} and 10^{-5} for every algorithm we tested. The displayed names correspond to the variants defined in Section 4. Moreover, they are compared with the result obatined by the *collaborative filtering* (CF) approach [1], which is the best result we are aware of. Figure 4 displays the distribution of the outputs of the GNNMean model. It is observed that the histogram resembles that of the training data set in Figure 3. However, the outputs are more concentrated on the scores 3 and 4, which are the most frequent ratings in the training data set and is the closest to the average score 3.53. On the other hand, such a situation is often observed in neural networks when the training data set is unbalanced and noise contaminated[11].

Table 3 show that FNNs perform just slightly worse than the corresponding GNNs. For example it is observed that GNNMean, which is based on user and movie average scores, performs as well as a FNN-Mean, while GNNNoData, which uses only the graphical structure information, performs poorly. Even if GNNs are universal approximators and can approximate any function on graphs [6], these experiments suggest that GNNs may not able to extract sufficient topological information from the graph to improve the performance. The reason of such a result is not completely clear, even though some hypotheses can be formulated. First, it is not clear whether there exists useful information that can be easily extracted from the structure of the graphs in order to minimize the MAE. The fact that our approaches and the collaborative filtering methods all obtain an average MAE that is close to 0.75 may give rise to the suspicion that the information needed to overcome such a limit is not easily obtained from the graph structures. GNNs are based on a local model of computation, such that a GNN can easily learn only functions of the network topology which involve nodes that are very close to each other. Unfortunately

[11] Here, the noise is due to the fact that users with similar features can rate the same movie with different scores.

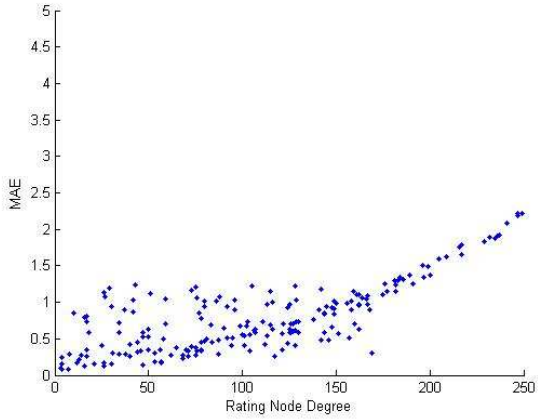


Fig. 4. Absolute Error value Vs. rating node degree for 200 test set samples

the dependencies among movies, ratings and users can be complex and could depend on global properties of the graph. Another problem may be the distribution of significant features in the given graphs. A GNN encodes all the information about the input graph into states associated with nodes (see Section 2). Since user degree and movie degree follow a power-law distribution, most of the nodes should be able to encode into their structural information coming from a massive number of directly connected nodes. Thus, due to the limitation of the dimension of the state such an encoding may not be possible in MovieLens graphs. Here, we have a combination of factors which might affect the learning process negatively, but the scale-free nature of the graphs seem to be one of the most important ones. Figure 4 suggests that this is an important phenomenon. Figure 4 plots the absolute error versus the rating node degree for 200 samples from the testing dataset. In this diagram, the absolute error with respect to GNNMean, and the rating node degree for node $s_{i,j}$ (defined as: $O(s_{i,j}) = \max\{O(u_i), O(m_j)\}$, where $O(m_j)$ is the number of user who rated movie M_j and $O(u_i)$ is the number of movies rated by user u_i) are shown. We also consider Table 3 with respect to the feature choice. For example, "Mean" is the simplest choice but other choices could be better if ratings play a central role in the prediction of the opinions of the users. The GNNMeanAndData, which extends GNNMean by the inclusion of specific information that describes the user and the movie with more details, performs somewhat worse than GNNMean. Such an observation suggests that the included information does not contribute to the prediction, but on the contrary, it introduces a noise that makes it slightly more difficult to learn the model. Such an observation is confirmed by the low performance of GNNOnlyData that uses only the information added in GNNMeanAndData with respect to GNNMean. Moreover, note that GNNDegree uses a feature set containing the number of movies rated by a given user and the number of users which have rated a given movie. In practice, those features allow us to measure, for example, the popularity of a movie. Here, it is interesting to note that, according to the results, such information is less useful for the prediction than the one used in GNNMean.

6 Conclusions and Future Work

This paper highlights some limitations of graph neural networks when applied to recommender systems. We tested the GNN model on the MovieLens dataset, and found that it performs as well as the classic FNN using the same feature set. We analysed how the scale-free nature of the involved graphs can negatively impact on the GNN performances. Future research topics include the experimentation of the GNN on other non-random graph topologies, in order to exploit training/test effects of particular topologies on the model.

7 Acknowledgments

The work presented in this paper received financial support from the Australian Research Council in form of a Linkage International Grant.

References

1. B. Sarwar, G. Karypis, J. Konstan and J. Riedl: *Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering,* Proceedings of the Fifth International Conference on Computer and Information Technology, 2002.
2. A. L. Barabasi and R. Albert: *Emergence of Scaling in Random Networks,* Science **286** (1999): 509–512.
3. L. Almeida: *A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment,* In: IEEE International Conference on Neural Networks, M. Caudill and C. Butler, Eds., vol. **2**, San Diego, 1987: IEEE, New York, 1987, pp. 609–618.
4. F. Pineda: *Generalization of Back–propagation to Recurrent Neural Networks,* Physical Review Letters, vol. **59**, pp. 2229–2232, 1987.
5. F. Fouss, A. Pirotte, M. Saerens: *A Novel Way of Computing Dissimilarities between Nodes of a Graph, with Application to Collaborative Filtering,* 15th European Conference on Machine Learning (ECML 2004); Proceedings of the Workshop on Statistical Approaches for Web Mining (SAWM), pp 26-37.
6. M. Gori, M. Hagenbuchner, F. Scarselli and A. C. Tsoi: *Graphical-based Learning Environment for Pattern Recognition,* In: Proceedings of SSPR 2004 (Syntactic and Structural Pattern Recognition), August 2004, invited paper.
7. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl: *Item-based collaborative filtering recommendation algorithms,* In: Proceedings of the 10th International World Wide Web Conference (WWW10), Hong Kong, May 2001.
8. U. Shardanand and P. Maes: *Social Information Filtering: Algorithms for Automating "Word of Mouth", In: Proceedings of CHI 95*, Denver.
9. J. Schafer, J. Konstan and J. Riedl: *Electronic commerce recommender applications,* Data Mining and Knowledge Discovery, Jan. 2001.
10. B. Miller, J. Riedl and J. Konstan. *GroupLens for Usenet: Experiences in applying collaborative filtering to a social information system,* In: C. Leug and D. Fisher, editors, From Usenet to CoWebs: Interacting with Social Information Spaces. Springer-Verlag, 2002.
11. K. Goldberg, T. Roeder, D. Gupta and C. Perkins: *Eigentaste: A constant time collaborative filtering algorithm,* Information Retrieval, **4**(2):133–151, 2001.
12. G. H. Golub, C. F. V. Loan: *Matrix computations,* (3nd ed.), Johns Hopkins University Press, Baltimore, MD, USA 1996.
13. L. Page, S. Brin, R. Motwani, T. Winograd: *The pagerank citation ranking: Bringing order to the web,* In: Proc. of ASIS '98, Pittsburgh, USA 1998.