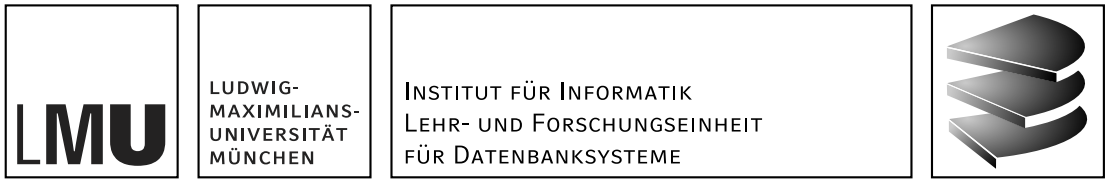


Report

in Seminar Recent Developments in Data Science

A General Graph-based Model for Recommendation in Event-based Social Networks

Tuan-Anh Nguyen Pham , Xutao Li , Gao Cong , Zhenjie Zhang
School of Computer Engineering, Nanyang Technological University, Singapore

Author:	Christian Lemke
Supervisors:	Julian Busch, Evgeniy Faerman, Felix Borutta
Institution:	Ludwig-Maximilian University of Munich
Field of study:	Master computer science
Place and date:	Munich, 08.06.2017

Contents

Introduction

1.1 Preconditions

This report is written as a part of the Seminar Recent Developments in Data Science. The goal of the seminar is to discuss recent concepts and approaches in the area of Data Science including data management and data analytics. The focus lies on the effective models and efficient technologies for large databases of complex objects. A written report, an own developed scenario of the topic and a presentation in front of the participants with discussion are the parts of the seminar. [dbs]

As this report is limited on 10 pages, it focuses on the description of the used model, its procedure and the proof with an own developed scenario.

1.2 Motivation and Recommendations systems

The amount of data is growing exponentially over the last decades. That leads us into a world with almost infinite abundance of information resulting in a huge amount of options. Barry Schwarz describes this problem in the Paradox of Choice. If there are too many options it would take a huge amount of time to pick one. It is also possible to get frustrated and ending up not picking anything at all or get the Fear of missing out (FOMO). [OHS09]

Recommendation systems can help to solve this problem. With Machine Learning techniques the systems can learn the users preferences and recommend options. Some examples are Google web search, Amazon product recommendations or Facebook friends recommendations. There are two ways how they work: One is the collaborative way which searches for similar users based on what they liked and create recommendations based on their choices. The second one is the content-based way. It predicts what the user likes based on what he liked in the past. Hybrids are also often in use.

1.3 Event-based Social Networks

An new field of social networks named Event-based Social Networks (EBSNs) is growing over the recent years. Its purpose is to bring people together and help them to organize social events. One EBSN example is meetup.com. It has currently over 30 million members, over 600 thousand events every month, is available in 182 countries and is still growing. [mee]

HeteRS

HeteRS stands for general-purpose Heterogeneous graph-based Recommendation System model. It is the model for recommendations described by the paper by Tuan-Anh Nguyen Pham et al..

First, an introduction is given. Next, a detailed description of how the model works follows. A parameter learning and approximation algorithm is described thereafter. At last, the experiments and their results are described.

2.1 Introduction

EBSNs offer several possibilities for recommendations. The network contains different types of entries and connections between them. Figure [1] illustrates an example of a heterogeneous graph. It can be seen that some types of nodes can have a connection to specific types of nodes and others not. For example, there are connections of user nodes to event nodes but not to venue nodes. The example shows, that users attended different events, belong to different groups and have different tags. For a recommendation system, this ensues in different recommendation possibilities. Tuan-Anh Nguyen Pham et al. focus on the following kinds of recommendations:

- Group-to-user
- Event-to-user
- Tag-to-group

To be noticed, this model is also capable for more recommendations. For example, tag-to-user or venues-to-events can be very useful als well, in my own opinion.

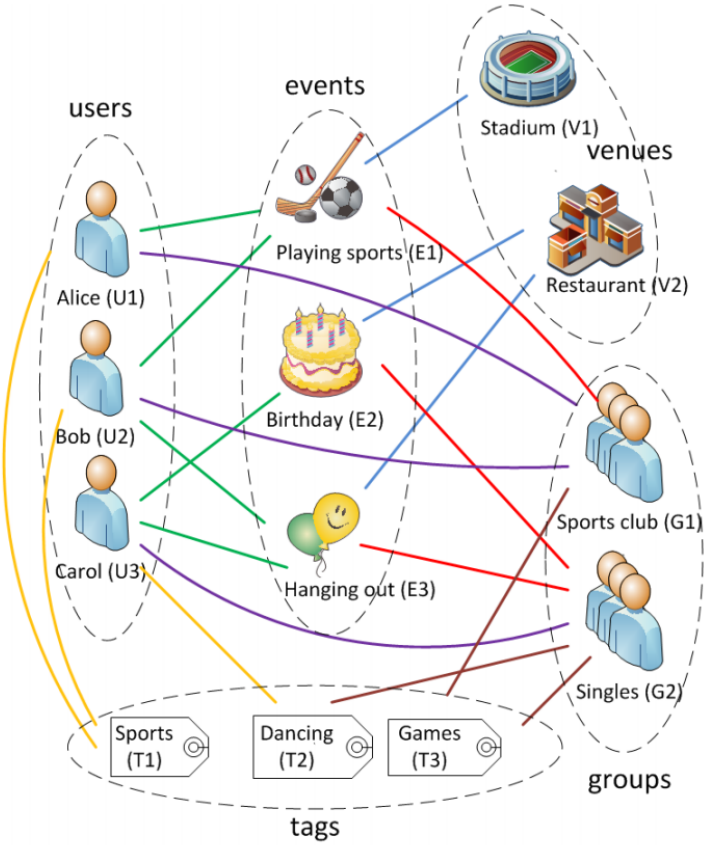


Figure 2.1: Graph of the EBSN usd in the paper. Source [PLCZ15]

A graph based model with different types of entries (a homogeneous graph) gives the advantage, that it is possible to assign different influence weights between the edges of different node types. This makes it possible to maintain more information. For example, an event-to-user recommendation of the graph of Figure 2.1 is able to maintain information not only from user and event nodes, but also from the influences from other nodes connected to them.

2.2 Basics and background

Random Walk with Restart (RWR) has been used for many graph-based recommendation systems. The problem is that RWR is based on univariate Markov chains with homogeneous graphs and our heterogeneous graph uses different types of nodes with different influences between them. The authors write in the chapter recent works, that several works project a heterogeneous to a homogeneous graph to work with RWR by treating all the nodes and edges

as the same type. Also some works used different weights for the edges on the projection. But the authors found out and showed, that this weights will lose their original meaning and may not play the expected roles. They explain the problem that RWR is not able to measure influence weights of different node types.

Multivariate Markov chain (MMC) works with heterogeneous graphs and is able to assign influence weights, or also named parameters, between different types of nodes. This leads to the problem of choosing the right influence weights. Tuan-Anh Nguyen Pham et al. came up with the idea of parameter learning in a MMC-based model. Of their best knowledge, they have been the first with this idea. MMC also leads into an efficiency problem which they encounter with an approximation problem. This three contributions will be described in the next paragraphs. [PLCZ15] [CHNS13]

2.3 Model

In the paper an heterogeneous MMC-based recommendation model is explained and exemplified on an ESBN-network with 6 different types of entries on a big dataset. To explain and understand such a MMC-model, we are certain that a small example is more helpful. For that see our model description of HeMoRS in chapter 3. A fully description of the HeteRS model would be an unnecessary repetition and would not help to explain an MMC-based model.

2.4 Temporal Factor

The time of an event is an important factor in the behavior how users join events. The histogram in figure 2.2 shows that users join events mostly on a weekly pattern. It is reasonable to model this behavior into the graph. For that the authors insert session nodes. For each weekday a user joins an event, a session node with the userid and the weekday is created and connected to the corresponding event. Figure 2.3 shows an example of connected session nodes in the graph. With the session nodes the users should get a higher recommendation on events of the weekdays they normally visit events.

The second observation from figure 2.2 is that the number of event pairs decreases as the duration increases. In other words, a user joins an event more likely if he or she joined one lately. So recent events are more important than earlier ones. The authors model this short-time period behavior by weighting the incoming edges of events by the decay function in function 2.1. It returns the weight of event j at time tm_j. It calculates the difference of days with

$tm_c - tm_j$, while tm_cis the last event in the dataset. The influence of the time difference is controlled by η. A smaller η will rank the recent events higher.

$$f_t(tm_j) = exp(-\eta(tm_c - tm_j)) \tag{2.1}$$

To insert the decay function into the graph all adjacency matrices, A_{NE} while E stands for the event and N for all nodes with connection to E, have to be upgraded with the weight of function $f_t(tm_j)$.

According to our own opinion, it is not recommendable to manipulate the adjacency matrices. A better way could be to add a vector with the node specific weights of the decay function to manipulate the weights of the adjacency matrices before the normalization to the transition matrices. This would also make it easier to add more weights to specific nodes.

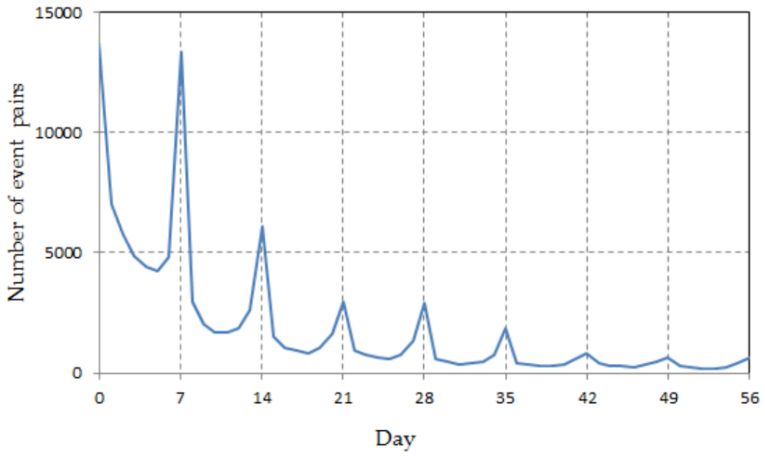


Figure 2.2: Duration between two consecutive events of users in New York City. Source [PLCZ15]

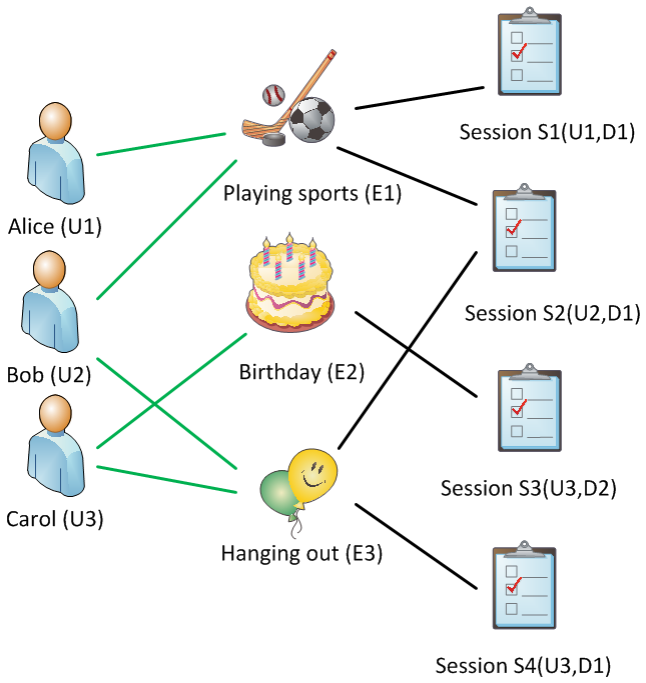


Figure 2.3: Connection of session nodes. Source [PLCZ15]

2.5 Optimization approach for parameter learning

From our equations in paragraph 2.3 we can see that the transition weights play an important role in finding good recommendations. Usually these parameters were set by hand. In this work, the optimal parameters were found out with the help of machine learning. This is, by the best knowledge of the authors, the first time somebody uses machine learning in a MMC-based model. In the following we will give an overview on the parameter learning approach and try to simplify it.

The Objective Function or also often called cost function uses AUC(Area Under the ROC Curve). In group-to-user recommendation, a given user u tries to rate groups that u joined higher, than groups he or she did not. These sets are denoted PG for positive groups and NG for negative groups. The function tries to maximize the ratio of $|PG|$ minus $|NG|$ to the sum of $|PG|$ and $|NG|$. The algorithm uses Sigmoid function to approximate $\mathbb{1}$ and uses stochastic gradient decent as an optimizer.

In other words, it tries several α values and calculates the resulting set of recommended groups. Then it compares the predicted groups with the real

joined groups of the user. This defines the predicted groups into PG, that u actually joined and NG that have been predicted wrong. For the next value it tries to maximize the ratio of $\frac{|PG| - |NG|}{|PG| + |NG|}$.
The used objective function is:

$$\max_{\alpha} Obj(\alpha) = \sum_{k=1}^{m} \frac{\sum_{i \in PG_k} \sum_{j \in NG_k} \mathbb{1}(g(i) - g(j))}{|PG_k| + |NG_k|} \tag{2.2}$$

where $\mathbb{1}$ is an indicator function that equals 1 if $g(i) > g(j)$ and 0 otherwise.

2.6 Approximation algorithm

The authors added an approximation algorithm to speed up the random walk based approach. For this the recommendation algorithm has to execute the model iteratively until the probability vectors converge. Time complexity depends on the number of edges in the graph and the increasing number of iterations that are needed to walk through all edges. To solve that problem, HeteRS uses local propagation. It focuses on the nearest neighbors of the query nodes by not allowing the iteration go too deep into the graph. This provides the algorithm to calculate unimportant nodes which would not have a big effect on the result. The authors could yield a much better efficiency without sacrificing much accuracy.

2.7 Experiments

In the following chapter, an overview about the experiment and evaluation of HeteRS is given. The used dataset, other baseline methods and the learned parameters are described. Because of space limitations, only the group-to-user recommendation results and their relevant parameters are described. The evaluation of the efficiency of the used approximation method is not part of this report.

2.7.1 Data overview and selection

For the dataset the authors crawled meetup.com and generated a graph for New York City (NYC) and for the state of California. As time range they used the year 2012 for both datasets. In the following, the NYC dataset is used, since both show similar results.

For the reprocessing of the group-to-user recommendation, an amount of groups of each user needs to be removed out of the graph to use them as a testset. In our opinion this leads to a problem, since many users joined only a few amount of groups, the resulting train- and testsets are unrepresentative small. The authors try to work against this problem by removing groups with less than 21 events and users with less than 6 participating events. This leads to removing a lot of data but increases the groups per users ratio. The whole procedure leads to a trainset shown in table 2.1.

Type	**value**
# Events	9,549
# Users	46,895
# Groups	398
# Tags	15,819
# Venues	2,396
Avg. Participants for an event	17.65
Avg. Events for a user	3.59
Avg. Members for a group	382.71
Avg. Groups joined by a user	3.16

Table 2.1: Statistics of used NYC Dataset

2.7.2 Baseline Methods

HeteRS is compared with state of the art methods for recommendation problems. In this report, we focus on a subset of methods. User-based Collaborative Filtering (CF), Bayesian Personalized Ranking (BPR) and PTARMIGAN are not described but their results are included.

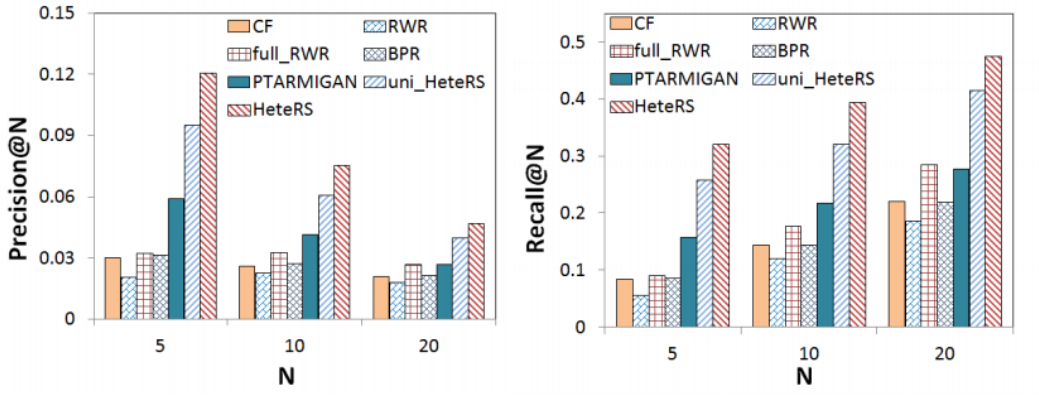
RWR: Random Walk with restart for group-to-user recommendation. Runs on a group-group interaction graph weighted by number of common users, where the groups of each test user are treated as query nodes.

full_RWR: performs RWR on our constructed heterogeneous graph and it treats the different types of edges (and nodes) the same way.

uni_HeteRS: This method is HeteRS without parameter learning part. Parameters are assigned with values uniformly. We use this variation of our method to evaluate the effectiveness of our learning process.

2.7.3 Results

As evaluation metric, precision p@N and recall r@N is used, where N is the size of the resultset. The result is shown in figure 2.4.



(a) Precision@N - NYC (b) Recall@N - NYC

Figure 2.4: Result of group-to-user recommendation. Shows precision and recall on N results. Source [PLCZ15]

HeteRS outperforms all the other base methods. It achieves better results than uni_HeteRS, which proves that the parameter learning approach is important and useful for better results.

The authors also tested HeteRS on tag-to-group and event-to-user recommendations. They used another dataset of Meetup.com of state california USA as well. Their results also outperformed state-of-the-art baseline methods.

2.7.4 Interpretation of Transition Parameters

This paragraph shows the results of the group-to-user parameter learning in table 2.2 and gives a description and interpretation of the gained values.

For a short recap of the meaning of the Transition weights(also called parameters), the α_{MN} means that N gets α_{MN} of its weights from M. The higher the weight, the more important is M for N in the recommendation.

We can see that G gets with 0.42 a high influence from U. This means we mostly rely on similar users when recommending groups to a user. Tags have with 0.31 a relatively high influence on groups. This leads to the assumption that users like to join groups with similar tags. So in other words users often join multiple groups with the same topic.

Transition Parameter	**value**
α_{GE}	0.38
α_{SE}	0.25
α_{UE}	0.12
α_{VE}	0.25
α_{EG}	0.27
α_{TG}	0.31
α_{UG}	0.42
α_{GT}	0.65
α_{UT}	0.35
α_{EU}	0.11
α_{GU}	0.11
α_{TU}	0.05
α_{q_uU}	0.73

Table 2.2: Learned transition parameters of the NYC dataset

HeMoRS

This chapter is about a self-made example of the examined topic. In this section we will present our self-made example HeMoRS.

3.1 Introduction and hypothesis

The topic of this work is a graph-based recommendations system with a heterogeneous graph using MMC with parameters learning and an approximation algorithm. In our scenario we want to explain and illustrate the model of HeteRS with a small and easy example. As data we created a small graph with users, movies and genres for a movie-to-user recommendation shown in figure 3.1. Parameter learning is not a part of this example, because it would not be reasonable with such less and own created data. There is also no need to run an approximation algorithm on that small size.

Our created graph shows a simple scenario with four users $u_1, \ldots, u_4$, four movies $m_1, \ldots, m_4$ and two genres g_1, g_2. In this example, u_4 is the users for the movie-to-user recommendation. The hypothesis is, that u_4 gets a recommendation of m_3 before one on m_4, because m_3 has the same genre g_1 as his only watched movie m_1 and m_4 does not. Proving this hypothesis shows that more different node types lead to more information and to a better recommendation result.

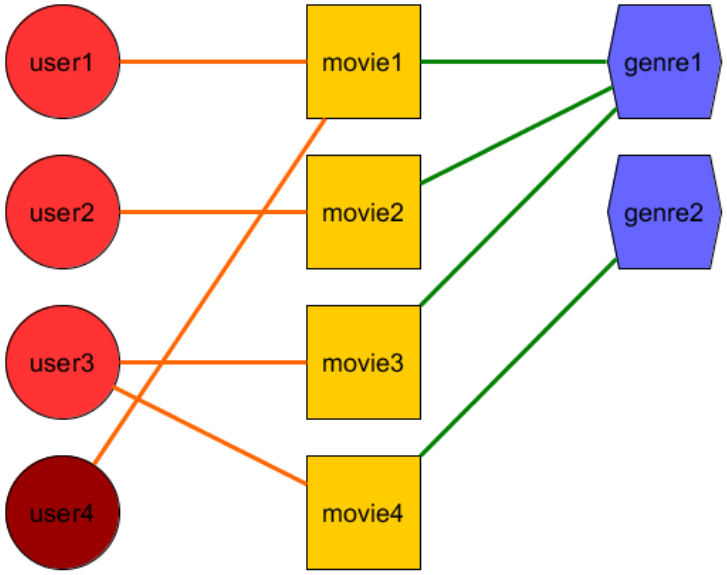


Figure 3.1: Graph of the movie-to-user recommendation example.

3.2 Variables

The following describes the different variables.

Transition matrices: The edges of the graph shown in figure 3.1 are represented by the transition matrices. They contain the information which type of node m is connected to type n. There are multiple transition matrices, one for each direction of related node types. The notation of the paper is different to the normally used notation of adjacency matrices. The authors mention their notation $P_{MN}(N, M)$ only in a sub sentence which led us to many misunderstandings and confusions. Notation $P_{MN}(N, M)$ means transition matrix P_{MN} contains the edges from M to N but the matrix representation uses N for their rows and M for their columns. Normally, in adjacency matrices its the other way around. The matrices get normalized by columns.

The equations 3.1 show the transition matrices for HeMoRS. Matrices 3.1a and 3.1b match the orange edges in figure 3.1 and represent the connections between the user and movie nodes. Matrices 3.1c and 3.1d are likewise to the green edges between movies and genres.

$$P_{UM} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 \end{pmatrix} \tag{3.1a}$$

$$P_{MU} = \begin{pmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0.5 & 0 & 0 & 0 \end{pmatrix} \tag{3.1b}$$

$$P_{GM} = \begin{pmatrix} 1/3 & 0 \\ 1/3 & 0 \\ 1/3 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.1c}$$

$$P_{MG} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.1d}$$

Parameters: α_{XY} is the transition weight (also called parameter) that node y gets from x. The parameters are set intuitively. Movies get 0.75 of their weight form users and 0.25 from genres. The probability for a teleport is set to 20%. The selected parameters are shown in table 3.2.

Note that this setting does not affect the order of the movie recommendation result for u_4, as long as none of the parameters are set to zero. The parameters would have an impact if we add for example an actor node type connected to the movie. Then the transition weight distribution from users genres and actors to movies would have an impact on the result by defining which node type should have more influence.

Transition Parameter	**value**
α_{UM}	0.75
α_{MU}	0.8
α_{MG}	1.0
α_{GM}	0.25
α_{UU}	0.2

Table 3.1: Selected parameters for HeMoRS

Query Vector q: Vector q leads to personalized results. It represents the

chance to teleport to u_4. That is how u_4 gets a higher weight to distribute to its closer neighbor nodes.

$$q = \begin{pmatrix} \alpha_{UU} * {q_u}^T \\ 0 * {q_m}^T \\ 0 * {q_g}^T \end{pmatrix} = \begin{pmatrix} \alpha_{UU} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.2}$$

Matrix M: It contains the parameters multiplied by the corresponding transition matrices.

$$M = \begin{pmatrix} 0 & \alpha_{MU} P_{MU} & 0 \\ \alpha_{UM} P_{UM} & 0 & \alpha_{GM} P_{GM} \\ 0 & \alpha_{MG} P_{MG} & 0 \end{pmatrix} \tag{3.3}$$

Probability distribution vector r: It represents the probabilities that users, movies and genres are visited at time t. During the iterations r represents the result of the iteration steps and converges to the recommendation result. Equation 3.4 shows the probabilities of nodes u, m, g of timestep t in the certain order.

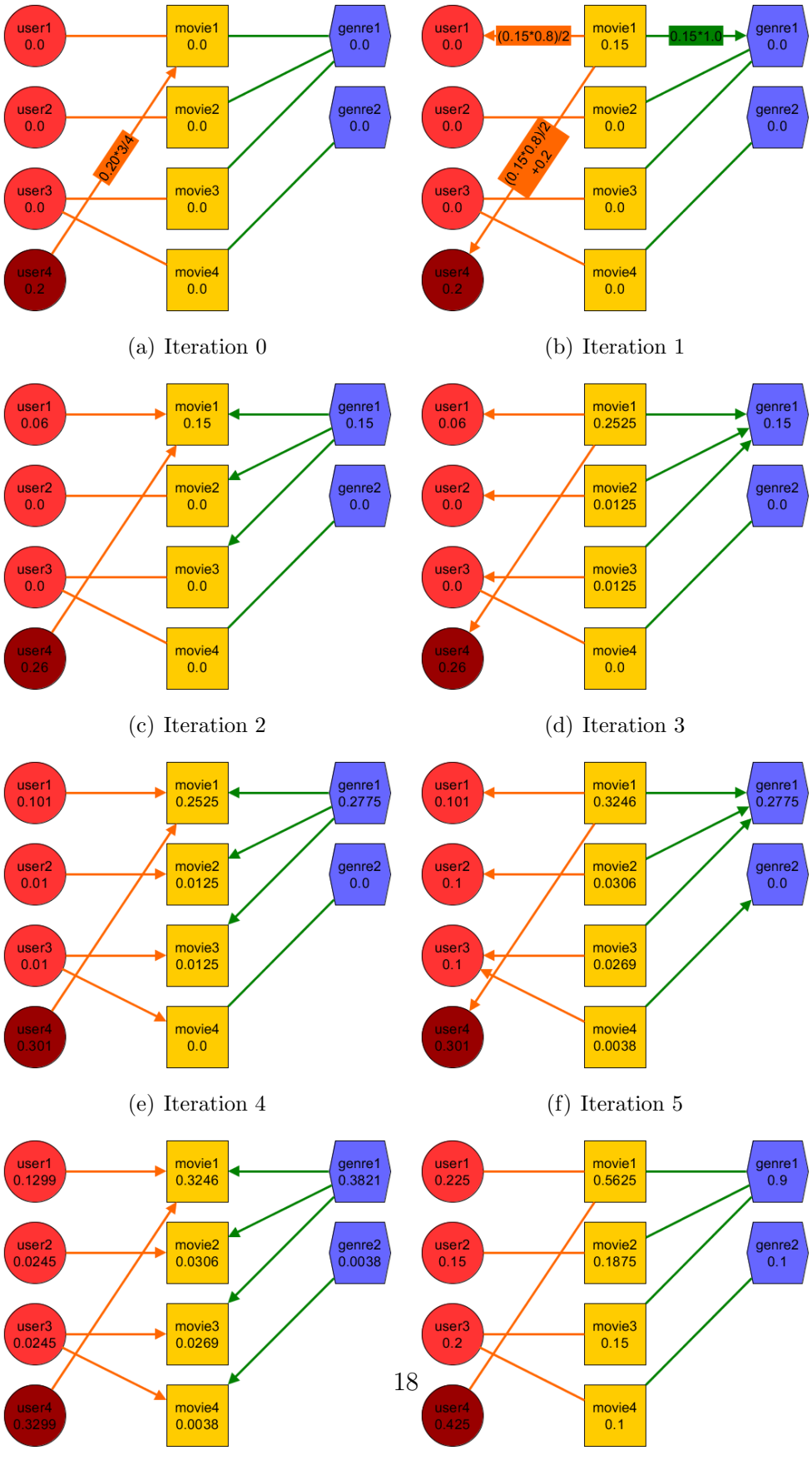
$$r^t = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ m_1 \\ m_2 \\ m_3 \\ m_4 \\ g_1 \\ g_2 \end{pmatrix} \tag{3.4}$$

3.3 Algorithm

In this paragraph we want to illustrate the iteration of the algorithm. Note that $r^{(t_0)}$ gets initialized by q for a better visualization of the weight distribution per iteration. This initialization is independent from the result, because r always converges to the result unless $r = 0$. Other initializations could give an earlier convergence. Equation 3.5 shows the algorithm that convergences r to the final result.

$$r^{(t+1)} = Mr^{(t)} + q \tag{3.5}$$

Figure 3.3 shows the iteration steps from the initialization to the iteration 6, when all nodes got an probability and the converged result at iteration 360. Figure 3.2(a) shows the status after the initialization. For the first step, the product of the probability of user u_4 and the probability of α_{UM} is transferred to m_1. The result of the probability of 0.15 can be seen in figure 3.2(b). From this movie node the value of m_1 gets multiplied by the parameters α_{MU}, α_{MG} and their result added to the value of their target nodes. This process continues until the result converges. We get the convergence at iteration 360 with a calculation accuracy of 6 decimal places.



(a) Iteration 0 (b) Iteration 1

(c) Iteration 2 (d) Iteration 3

(e) Iteration 4 (f) Iteration 5

(g) Iteration 6 (h) Iteration 360

Figure 3.2: Iteration 0,. . . ,6 and 360 of HeMoRS

3.4 Result

At Iteration 360 r converged to a solid result by rounding with seven decimals. On such small example it is possible to check our result by solving linear equation 3.6 directly. This proves that our result is correct. Equation 3.7 shows the final result. It shows that the weights of the movies give a descending order of m_1, m_2, m_3 and m_4. So it shows that m_3 gets with 0.15 a higher weight than m_4 with 0.1, which proves our hypothesis from section 3.1.

$$r^* = (I - M)^{-1} * q \tag{3.6}$$

$$r^* = \begin{pmatrix} 0.225 \\ 0.15 \\ 0.2 \\ 0.425 \\ 0.5625 \\ 0.1875 \\ 0.15 \\ 0.1 \\ 0.9 \\ 0.1 \end{pmatrix} \tag{3.7}$$

Summary

This report gives an introduction on graph-based recommendation systems and EBSN's. We introduced the HeteRS-Model from Pham et al.. We have also given insights in several components of HeteRS. We tested the reproduceibility with an own small and easy understandable example. A movie-to-user recommendation scenario with additional genre information was created. With the help of this scenario and an easy to verify hypothesis, we were able to proof its comprehensibility. Visualizing the node probabilities of each iteration helped to understand the random walk approach. This chapter describes our achievements and experiences on the paper and our own work on it.

4.1 Critical assessment

We have some suggestions that would help for a better understanding of the paper and possible optimizations.

Time factor: The manipulation of the temporal factor on the adjacency matrices could be optimized. As already mentioned in section 2.1, the temporal factor manipulating the adjacency matrices for weighting each node independently is not recommendable in our opinion. The direct manipulation should be avoided. For example there could be an own time weight vector that gets multiplied on the adjacency matrices. This would lead to a better understanding of the influences of the time weights. We also noticed that the consequences or the procedure of directly manipulating adjacency matrices should be more described.

Data for evaluation: The authors filtered important data out of the training set. As already described in section 2.7.1, groups with less than 21 events and users with less than 6 participating events got removed. This is necessary to be able to split joined groups of users into training and testsets. But this also completely ignores users and groups which focus their activity on less events. Also new users or new groups get fully ignored by the algorithm.

Confusing matrix notation: As already mentioned in section 3.2 the authors used an unusual notation for their matrices. This led to misunderstandings and misleading assumptions. The reproducibility could be improved if the notation would be better described or if a common notation would be used.

Strengths and weaknesses: Overall, the paper is good understandable. Especially the introduction of the topic is well written. The transition to the model and the matrix notation could be improved. The paper shows how powerful and easy to use MMC-based approaches can be. Especially the invention of learning the transition parameters leads to a great improvement in the area of MMC-based approaches.

List of Figures

List of Tables

Bibliography

[CHNS13] Wai-Ki Ching, Ximin Huang, Michael K Ng, and Tak Kuen Siu. *Markov chains: models, algorithms and applications*, volume 189. Springer Science & Business Media, 2013.

[dbs] dbs. *Hauptseminar Recent Developments in Data Science, http://fogo.dbs.ifi.lmu.de/cms/Hauptseminar Recent Developments in Data Science SS17*. Accessed: 2017-06-30.

[mee] meetup. *About Meetup, http://meetup.com/about*. Accessed: 2017-06-30.

[OHS09] Antti Oulasvirta, Janne P Hukkinen, and Barry Schwartz. *When more is less: the paradox of choice in search engine use*. In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, pages 516–523. ACM, 2009.

[PLCZ15] Tuan-Anh Nguyen Pham, Xutao Li, Gao Cong, and Zhenjie Zhang. *A general graph-based model for recommendation in event-based social networks*. In Data Engineering (ICDE), 2015 IEEE 31st International Conference on, pages 567–578. IEEE, 2015.