

Recommendation on Item Graphs

Fei Wang

Department of Automation
Tsinghua University
Beijing, 100084, P.R.China
feiwang03@gmail.com

Sheng Ma, Liuzhong Yang

Vivido Media (Beijing) Inc.
Shangdi Development Zone
Beijing 100085, China
masheng@vividomedia.com.cn

Tao Li

School of Computer Science
Florida International University
Miami, FL 33199
taoli@cs.fu.edu

Abstract

A novel scheme for item-based recommendation is proposed in this paper. In our framework, the items are described by an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. \mathcal{V} is the node set which is identical to the item set, and \mathcal{E} is the edge set. Associate with each edge $e_{ij} \in \mathcal{E}$ is a weight $w_{ij} \geq 0$, which represents similarity between items i and j . Without the loss of generality, we assume that any user's ratings to the items should be sufficiently smooth with respect to the intrinsic structure of the items, i.e., a user should give similar ratings to similar items. A simple algorithm is presented to achieve such a "smooth" solution. Encouraging experimental results are provided to show the effectiveness of our method.

1. Introduction

The explosive growth of the world-wide-web and the emergence of e-commerce has led to the development of recommender system - a personalized information filtering technology used to identify a set of items that will be of interest to a certain user. In recent years, recommender systems have been used in a number of different applications [2, 5, 6, 8], such as recommending products a customer will most likely buy movies [15] or books [6].

Various approaches for recommender systems have been developed that utilize either demographic, content, or historical information [1, 8, 11]. Among these methods, *user based collaborative filtering* is perhaps the most successful popular one for building recommender systems to date [1, 8]. However, despite their success, there are still some major limitations in *user based recommendation algorithms*, such as *sparsity* and *scalability* [9, 11].

To overcome these problems, an alternate approach, named *item based recommendation*, is proposed [2, 6, 10]. In these approaches, the historical information is analyzed to identify relations between pairwise items, such that the purchase of an item (or a set of items) often leads to the purchase of another similar item (or a set of similar items). These approaches can use a pre-computed model, i.e., they

are capable of exploring the relationships between pairwise items *off-line*, and the recommendation for an *active user* only needs to find the items that are similar to other items the user has liked. Many researchers have shown that *item based recommendation algorithms* can recommend a set of items more quickly, with the recommendation results comparable to traditional *user based recommender systems*.

As another hot research area, structured data mining, such as graphs [12] and relational databases [3], has aroused considerable interests in machine learning and data mining community. We believe that most of the real world data have their own structures, as well as the item vectors¹ in recommendation systems. For example, in a movie recommendation system, different movies have different genres, and will be interested in different groups of people. As a result, in the data space spanned by these *movie vectors*, the movies of different genres may reside in different places. The movies of same genre may be more compact than the movies of different genres.

Based on such intuition, we propose a novel recommendation scheme based on *item graphs*. In our method, the item vectors are described by an undirected weighted graph. The nodes of the graph are the items, and the edges represent the pairwise item relationships. Associated with each edge is a *nonnegative* weight which encodes the pairwise item similarities. We believe that this *item graph* can reflect the intrinsic structure of the item vectors. Therefore, the ratings of a user on this item graph should be sufficiently *smooth*, i.e. it is more likely that a user will give similar ratings to similar items. This is also a common principle for *item based recommendation*.

In this paper, the *smoothness* of a user's ratings over the whole item graph can be efficiently computed through the *combinatorial graph Laplacian* of the item graph [12]. Then the predicted ratings of a user to his unrated items can be solved by minimizing this smoothness. Finally the items can be ordered by their predicted ratings and the top n ones

¹An item vector is a column vector with its i -th entry equal to the rating of the i -th user to this item.

will be recommended to the user. Our method shares the advantage of traditional *item based recommendation methods* that the *on-line* computational burden is very small. We can show that if there are N items, then the *on-line* computational complexity of our method is only $O(N)$.

The rest of this paper is organized as follows: section 2 will introduce some notations and related works. We will formally present our algorithm in section 3. The experimental results will be provided in section 4, followed by the conclusions and discussions in section 5.

2. Notations and Related Works

In a typical recommendation system, there is a set of users $\mathcal{U} = \{u_1, u_2, \dots, u_M\}$ and a set of items $\mathcal{I} = \{i_1, i_2, \dots, i_N\}$. And we can construct an $M \times N$ *user-item* matrix \mathbf{R} , with its (p, q) -th entry R_{pq} equal to the rating of the user u_p to the item i_q . If user u_p has not rated for item i_q , then $R_{pq} = 0$. Note that these ratings may either ordinal (as in *MovieLens*[15]) or continuous (as in *Jester*[4]). We use \mathbf{u}_p to denote the p -th row of \mathbf{R} , which is called the *user vector* of u_p , and \mathbf{i}_q to denote the q -th column of \mathbf{R} , which is called the *item vector* of i_q .

A typical item based recommendation algorithm contains two main phases, the *model building phase* and the *prediction phase*. In the model building phase, the similarities between each pair of items are computed and for each particular item i_p , the algorithm will store its k most similar items $\{i_p^1, i_p^2, \dots, i_p^k\}$ and their similarity values with i_p . Two typical similarity calculation methods include the *cosine similarity* and *conditional probability* [2].

More concretely, the *cosine similarity* between a pair of item vectors \mathbf{i}_p and \mathbf{i}_q can be computed by

$$\text{sim}(\mathbf{i}_p, \mathbf{i}_q) = \cos(\mathbf{i}_p, \mathbf{i}_q) = \frac{\mathbf{i}_p \cdot \mathbf{i}_q}{\|\mathbf{i}_p\| \|\mathbf{i}_q\|}, \quad (1)$$

where ‘ \cdot ’ denotes the vector *dot-product operation* and $\|\cdot\|$ represents the *Euclidean norm*.

Another way of computing the similarity between i_p and i_q is to use the *conditional probability* of purchasing one item based on the other has already been purchased. In particular, the conditional probability of purchasing i_p given that i_q has already been purchased $P(i_p|i_q)$ is

$$P(i_p|i_q) = \frac{\#(i_p, i_q)}{\#(i_q)}, \quad (2)$$

where $\#(\mathcal{X})$ represents the number of customers that have purchased the items in set \mathcal{X} .

After the model building phase, the item based methods have to do item recommendation in the prediction phase. Particularly, for an active user u_p who has a list of rated items \mathcal{U}_p , the algorithm first selects the set \mathcal{C} of candidate recommended items by taking the union of the k most similar items for each $i \in \mathcal{U}_p$, and removing the items those are already in \mathcal{U}_p . Then, for each $c \in \mathcal{C}$ the algorithm computes its similarity between \mathcal{U}_p as the *sum* of the similarities

between all the items $i \in \mathcal{U}_p$ and c , using only the k most similar items for each $i \in \mathcal{U}_p$. At last, the items in \mathcal{C} are sorted in a non-increasing order according to that similarity, and the top ones are selected as the recommended items.

Deshpane et al. [2] noted that the calculated item similarities may have different scales, which means that the similarities between each item $i \in \mathcal{I}$ and its k most similar items may be significantly different. From a statistical viewpoint, the *distribution* of the items may have different densities in different localities. As a result, if we treat each item *equally* when computing the similarities, the recommendation results may be somewhat biased. Therefore they proposed to normalize the similarities between each item $i \in \mathcal{I}$ and its k most similar items so that they can add up to one. And their experiments show that this approach can improve the quality recommendation results.

3. The Algorithm

We have reviewed some major user based and item based recommendation algorithm in last section. In this section we will introduce a new *item based* algorithm, called *Item Rating Smoothness Maximization (IRSM)*.

3.1. Item Graph

Like traditional item based methods, *IRSM* also contains the *model building phase* and the *prediction phase*. In the model building phase, *IRSM* describe the items and their relationships by an item graph, which is defined as follows.

Definition 1 (Item graph). An item graph is an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where

- (1) $\mathcal{V} = \mathcal{I}$ is the node set (\mathcal{I} is the item set, which means that each item is regarded as a node on the graph \mathcal{G});
- (2) \mathcal{E} is the edge set. Associated with each edge $e_{pq} \in \mathcal{E}$ is a weight w_{pq} subject to $w_{pq} \geq 0$, $w_{pq} = w_{qp}$.

In this paper, we propose to compute w_{pq} by

$$w_{pq} = \exp\{-\beta(1 - \cos(\mathbf{i}_p, \mathbf{i}_q))\}, \quad (3)$$

where β is a free parameter, and $\cos(\mathbf{i}_p, \mathbf{i}_q)$ is computed by Eq.(1). Intuitively, w_{pq} reflects the similarity between i_p and i_q . Therefore, we can construct an $N \times N$ (N is the number of items) similarity matrix \mathbf{W} , with its (p, q) -th entry

$$\mathbf{W}_{pq} = w_{pq}. \quad (4)$$

For acceleration considerations, we can sparsify \mathbf{W} by just keeping similarities between each item $i \in \mathcal{I}$ and its the k most similar items, and setting the similarities between i and the rest items to be zero, *i.e.*

$$\mathbf{W}_{pq} = \begin{cases} w_{pq}, & i_q \in \mathcal{K}(i_p) \text{ or } i_p \in \mathcal{K}(i_q) \\ 0, & i_q \notin \mathcal{K}(i_p) \end{cases} \quad (5)$$

Here $\mathcal{K}(i_p)$ and $\mathcal{K}(i_q)$ represents the set that contains the k most similar items of i_p and i_q . In this way, we can make the similarity matrix \mathbf{W} sparser so that the computations in latter steps can be faster.

3.2. Rating Smoothness

As stated in the introduction, the basic assumption behind *IRSM* is that a user's ratings should be sufficiently smooth with respect to the intrinsic structure of the items. That is, a user tends to give similar ratings on similar items. Here the *similar items* is defined as

Definition 2 (Similar items). An item i_p is said to be similar to i_q if

- (1) $\text{sim}(\mathbf{i}_p, \mathbf{i}_q) \geq \delta$
 - (2) $\text{sim}(\mathbf{i}_p, \mathbf{i}_r) \geq \delta$, and $\text{sim}(\mathbf{i}_q, \mathbf{i}_r) \geq \delta$
- where $\delta > 0$ is a predefined threshold, \mathbf{i}_p , \mathbf{i}_q , \mathbf{i}_r are the corresponding item vectors of i_p , i_q , i_r .

On an item graph, two items are similar means that: (1) if two item nodes are connected by an edge with a large weight, then the two items are similar; (2) if there exists a path connecting two item nodes, and the weights on each edge that constitutes this path are all sufficiently large, or, from a statistical viewpoint, the regions that the path goes through have high density, then the two items are similar.

Therefore, the item graph reflects the intrinsic structure of the item data. And the basic assumption of *IRSM* can be re-expressed as to maximize the *smoothness* of the item rating of an active user over the whole item graph. According to [13], such *smoothness* can be measured by

$$\begin{aligned} \mathcal{S}(f) &= \sum_{i_p, i_q \in \mathcal{I}} w_{pq} \left(\frac{1}{\sqrt{d(i_p)}} f(i_p) - \frac{1}{\sqrt{d(i_q)}} f(i_q) \right)^2 \\ &= \mathbf{f}^T (\mathbf{I} - \mathbf{S}) \mathbf{f}, \end{aligned} \quad (6)$$

where $d(i_p) = \sum_q w_{pq}$, $\mathbf{S}_{pq} = w_{pq} / \sqrt{d(i_p)d(i_q)}$.

Now let's return to our recommendation task. The active user is u_l , with \mathbf{u}_l being its user vector. If we treat the elements in \mathbf{u}_l as the values returned by a rating function defined on the item graph, then our goal is to predict missing values in \mathbf{u}_l , and we can achieve it by minimizing the smoothness of \mathbf{u}_l over the item graph, *i.e.*,

$$\begin{aligned} \text{minimize} \quad & E(\mathbf{u}_l) = \mathbf{u}_l^T (\mathbf{I} - \mathbf{S}) \mathbf{u}_l \\ \text{s.t.} \quad & \mathbf{u}_l(\text{rated}) = r(\text{rated}) \end{aligned} \quad (7)$$

where the constraint is a boundary condition stating that we should keep the ratings that have already been given by u_l unchanged. In such a way, the recommendation problem is formulated as a constrained optimization problem. Using the same procedure as in [14], we can get

$$\mathbf{u}_{lu} = -\mathbf{L}_{uu}^{-1} \mathbf{L}_{ul} \mathbf{u}_{lr} \quad (8)$$

3.3. From Hard Constraint to Soft Constraint

A potential problem of the above algorithm is that we need to compute \mathbf{L}_{uu}^{-1} for each *active user*. Since different user rates for different items, this matrix inverse computation should be computed on-line, which is impractical when the number of items is very large. Therefore, we propose to

change the hard constraint in Eq.(7) to a soft one through introducing a regularization parameter $\gamma > 0$, *i.e.*

$$E'(\mathbf{u}_l) = \mathbf{u}_l^T (\mathbf{I} - \mathbf{S}) \mathbf{u}_l + \gamma \|\mathbf{u}_l - \mathbf{r}_l\|^2 \quad (9)$$

Here \mathbf{r}_l is the initial rating vector with its p -th entry

$$r_{lp} = \begin{cases} \text{the rating } u_l \text{ give to } i_p, & \text{if } i_p \text{ is rated by } u_l \\ 0, & \text{if } i_p \text{ is unrated} \end{cases}$$

And we can solve \mathbf{u}_l by set

$$\partial E'(\mathbf{u}_l) / \partial \mathbf{u}_l = 0 \implies \mathbf{u}_l = (1 - \alpha) (\mathbf{I} - \alpha \mathbf{S})^{-1} \mathbf{r}_l, \quad (11)$$

where $\alpha = 1/(1 + \gamma)$, and \mathbf{I} is an $N \times N$ identity matrix. An issue should be addressed here is that using such a variant the predicted ratings of the rated items may be different from the original ratings that u_l gives, but this is reasonable since these ratings may contain noises.

Using Eq.(11), we can see that we only need to compute the inverse of the matrix $\mathbf{I} - \alpha \mathbf{S}$, which is equal for all the users. Therefore we can perform this calculation off-line, *i.e.* we only need to compute a matrix-vector multiplication on-line, whose computational complexity is $O(N)$. The basic flowchart of our *IRSM* algorithm is shown in Fig.1.

4. Experiments

In this section we experimentally evaluate the performance of our *IRSM* recommendation algorithm and compare it with that of the traditional item-based recommendation algorithms. Three datasets are used in our experiments, namely *movielens*[15], *eachmovie*[16] and *Jester*[4].

4.1. Evaluation Metric

There have been many evaluation metrics for recommendation algorithms, such as *recall*[2], *MAE*, *MSE*[10], *NMAE*[7]. However, since in this paper what we want is to rank the unrated data and recommend the top ones to the active user, we need not to compute very accurate ratings, but we want the predicted preferences (or, equivalently, the order of the unrated items) to be accurate. Therefore we define a novel measure *Order Consistency (OC)* to measure how identical the predicted order to the true order.

Definition 5 (Order Consistency). Assuming there are d items, \mathbf{a} is the vector that these d items are sorted in an decreasing order according to their predicted ranking scores by *IRSM*, \mathbf{b} is the vector that these d items are sorted in a decreasing order according to their true ratings. For these d items, we have $C_d^2 = d!/(2!(d-2)!)$ ways to randomly select a pair of different items. \mathcal{A} is the set whose elements are pairwise items whose order in \mathbf{a} are the same as in \mathbf{b} , then order consistency (*OC*) is defined as

$$OC = |\mathcal{A}| / C_d^2, \quad (12)$$

where $|\mathcal{A}|$ represents the cardinality of \mathcal{A} .

From the above definition we can see that what *order consistency* reflects is that how identical that the order of the d items in \mathbf{a} to the order of them in \mathbf{b} . The more the value of *OC* to one, the better the prediction results are.

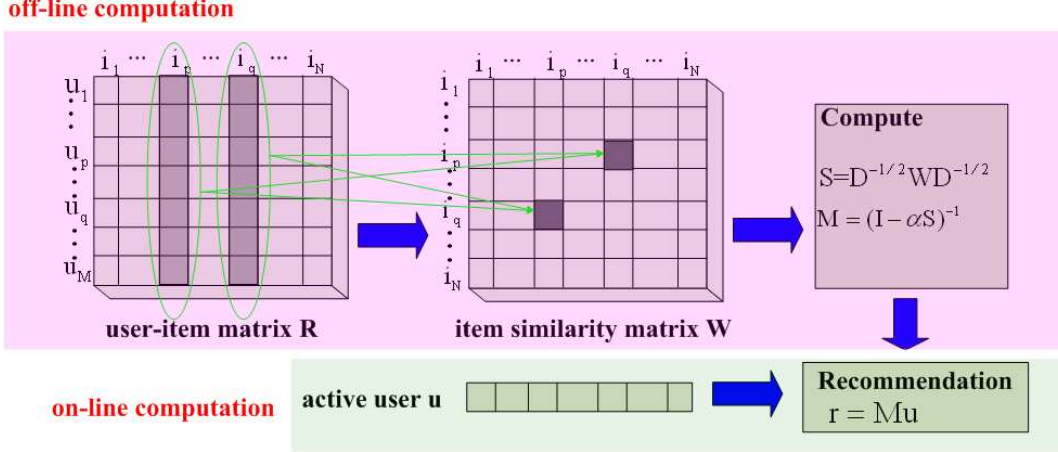


Figure 1. Flowchart of the *IRSM* algorithm.

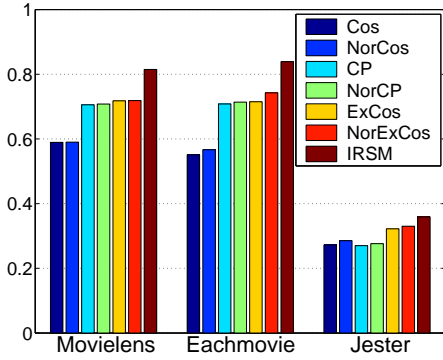


Figure 2. Performance comparison of different methods. The ordinate represents the order consistency value.

4.2. Comparison with Other Item Based Methods

We compare the performance of our *IRSM* method with the traditional item based method using different similarity measures, namely the *cosine* (Cos), *conditional probability* (CP) and *exponential cosine* (ExCos) similarity measures, where the *exponential cosine* is defined as in Eq.(3). Moreover, we also test the performances of these methods after similarity normalization (for details see [2]).

In *IRSM* and *ExCos* similarity based methods, the free parameter β in Eq.(3) are set by a 5-fold *cross-validation* way. And in *IRSM*, the *regularization parameter* γ (see Eq.(11)) is set to 1 manually. The results are shown in Fig.2, where the *OC* values are averaged over all the tested users. From the figure we can clearly see the advantage of our *IRSM* algorithm, and the similarity normalization can indeed improve the final recommendation results.

One issue should be addressed here is that in our *IRSM*

method, we just use the fully connected item graph in our experiments, that is, we do not sparsify the weight matrix as in Eq.(5), since we think that sparsify W may lose some similarity information. However, we find that sparsify W will not affect the final results significantly, and this part of experiment will be provided in section 4.4.

4.3. The Influence of γ

In this subsection we will discuss the influence of the regularization parameter γ in Eq.(9) to the final recommendation results in our *IRSM* method. Intuitively, γ reflects the tradeoff between the importance of the data geometry and the initial rating information. In our experiments, we vary $\alpha = 1/(1 + \gamma)$ from 0.1 to 1, and test the final *order consistency* values for *IRSM* on all the three datasets (since when $\alpha = 0$, then $u_i = r_i$, which means that we cannot achieve any recommendation at all, so we impose α to start from 0.1). The results can be seen in Fig.3.

In Fig.3, the ordinate represents the *OC* values, which are averaged over all the users. And we also plot the variances of them as error bars. The abscissa is the α values. From the figure we can clearly see that when $\alpha = 1$ (which means $\gamma = 0$), the *OC* value have a sudden drop, which means that we should consider both the geometry of the item data and the initial rating values. However, when α varies from 0.1 to 0.9, it seems that the *OC* values will not change significantly, *i.e.*, the recommendation results is not sensitive to the value of the regularization parameter.

4.4. The Influence of the Neighborhood Size

As we stated in section 4.2, in previous experiments, we just use the *fully connected* item graph. But this needs more storage requirements. Moreover, since we need to compute the inverse of matrix $I - \alpha S$, if we can make W sparser, then

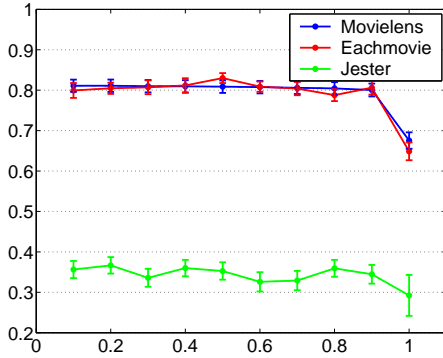


Figure 3. The influence of the regularization parameter. The abscissa is $\alpha = 1/(1 + \gamma)$, and the ordinate is the *OC* value.

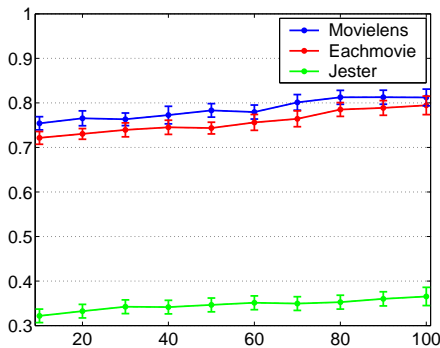


Figure 4. Influence of the neighborhood size. The abscissa is the size of the neighborhood, and the ordinate is the *OC* value.

many acceleration techniques can be applied to accelerate the matrix inverse computation. So we propose to sparsify \mathbf{W} by Eq.(5). But a natural question is whether this sparsification will affect the final recommendation results. In this section we will have a discussion on it.

Fig.4 shows our experimental results, in which we vary the number of nearest neighbors k for each item from 10 to 100 when sparsifying \mathbf{W} . From the figure we find that the neighborhood size will not affect the final prediction results significantly, so that we can use the sparsified \mathbf{W} in our *IRSM* algorithm which will alleviate some off-line computational burden.

5. Conclusions

In this paper we propose a novel item based recommendation method called *Item Rating Smoothness Maximization (IRSM)*. Unlike traditional item based methods which are based some intuitions, our method can explore the geometric information of the item data and make use of these information to produce better recommendations. Both theoretical analysis and experimental results are presented to show the effectiveness of our method.

Acknowledgments

The work of Tao Li is partially supported by the National Science Foundation under Grants IIS-0546280 and HRD-0317692.

References

- [1] Breese, J. S., Heckerman, D., Kadie, C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *UAI-1998*.
- [2] Deshpande, M., Karypis, G. Item-Based Top-N Recommendation Algorithms. *ACM Trans. On Information Systems-2004*.
- [3] Dzeroski, S. Multi-Relational Data Mining: An Introduction. *ACM SIGKDD Explorations Newsletter-2003*.
- [4] Goldberg, K., Roeder, T., Gupta, D., Perkins, C. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval-2001*.
- [5] Herlocker, J., Konstan, J., Borchers, A., Riedl, J. An Algorithmic Framework for Performing Collaborative Filtering. *SIGIR-1999*.
- [6] Linden, G., Smith, B., York, J. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing-2003*.
- [7] Marlin, B. Collaborative Filtering: A Machine Learning Perspective. Phd thesis. University of Toronto. 2004.
- [8] Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P., Riedl, J. 1994. Grouplens: An Open Architecture for Collaborative Filtering of Netnews. *SIGSCW-1994*.
- [9] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. Application of Dimensionality Reduction in Recommender System – A Case Study. *ACM WebKDD-2000*.
- [10] Sarwar, B. M., Karypis, G., Konstan, J., Riedl, J. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. *WWW-2001*.
- [11] Schafer, J., Konstan, J., Riedl, J. Recommender systems in e-commerce. In *Proceedings of ACM E-Commerce-1999*.
- [12] Wilson, R. C., Hancock, E. R., Luo, B. Pattern Vectors from Algebraic Graph Theory. *IEEE Trans. on Pattern Analysis and Machine Intelligence-2005*.
- [13] Zhou, D., B. Schölkopf.: Regularization on Discrete Spaces. *Pattern Recognition. Proceedings of the 27th DAGM Symposium*, 361-368, Springer, Berlin, Germany-2005.
- [14] Zhu, X., Ghahramani, Z., Lafferty, J. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the 20th ICML*, 2003.
- [15] <http://movielens.umn.edu>
- [16] <http://research.compaq.com/SRC/eachmovie/>