Movie Recommendations based in explicit and implicit features extracted from the *Filmtipset* dataset

Fernando Díez[1], J. Enrique Chavarriaga[1], Pedro G. Campos[1,2], Alejandro Bellogín[1]
[1]Universidad Autónoma de Madrid
Francisco Tomás y Valiente 11
28049 Madrid, Spain
+34 91 4972213
[2]Universidad del Bío-Bío
Av. Collao 1202
Concepción, Chile
{fernando.diez, alejandro.bellogin}@uam.es, {jes.chavarriag, pedro.campos}@estudiante.uam.es

ABSTRACT
In this paper, we describe the experiments conducted by the *Information Retrieval Group* at the Universidad Autónoma de Madrid (Spain) in order to better recommend movies for the 2010 CAMRa Challenge edition. Experiments were carried out on the dataset corresponding to social *Filmtipset* track. To obtain the movies recommendations we have used different algorithms based on Random Walks, which are well documented in the literature of collaborative recommendation. We have also included a new proposal in one of the algorithms in order to get better results. The results obtained have been computed by means of the trec_eval standard NIST evaluation procedure.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – Information Filtering, Retrieval Models, Selection Process; I.5.1 [Pattern Recognition] - Models

General Terms
Algorithms, Performance.

Keywords
Recommender Systems, Movie Recommendations, Social Networks, Random Walk, Challenge.

1. INTRODUCTION
There is a big growth of systems offering a vast quantity of data by means of social relations. Systems are, day by day, more and more specialized. Nowadays technological advances allow the technology based on social relations to grow continuously. There are numerous information based systems in which users can share different type of information on products and / or services. Users establish rich relations among them based on the new technological paradigms. Up to now recommender systems have been based primarily on implicit relationships between users and items they recommend. Recommender Systems (RS) have exploited this type of relationship to a high degree of sophistication and efficiency, as we can observe in sites like Amazon or Netflix. This successful behavior has led researchers and software developers to consider new forms of social relations between individuals. It is increasingly common to find systems where, for example, friendship relations between people are explicit, as in the case of the datasets used in the CAMRa Challenge [1]. Therefore, the current trend is strengthening in specialization and customization. The objective is to enhance the quality of the recommendations, without undermining the systems effectiveness. The success derived from the widespread use of social networking poses new problems in RS research. In the following, we mention some of the main problems found in these systems:

- Scalability issues [2]. Recommendations are inferred (sometimes weekly, or even daily), from huge amounts of data, as part of the system behavior.
- Reliability of the recommendations made by the users [3]. There are malicious users ratings introduced in order to damage the performance of the system itself.
- Compromising information. Due to privacy reasons not all systems allow access to explicit relations between groups of people. The existence of datasets like Movielens, Last.fm, Moviepilot or Filmtipset is crucial in order to improve the performance of the recommendation algorithms.

The remainder of the paper is structured as follows: section 2 presents related work on Collaborative Filtering (CF) and social RS. Section 3 describes the experiments performed for the Challenge. Section 4 discusses the results obtained in the experimentation. Finally, section 5 presents some preliminary conclusions and expected future work.

2. RELATED WORK
2.1 Collaborative Recommendation
There are several areas involved in the process of information modeling in RS. Some of these areas are, for example, the study of the variability in the users ratings [4], the users coverage of the dataset [5], information provided by external users or experts [6], the temporal dimension [7], [8], [9] and the use of spatial-temporal information [3]. Frequently there exist limited data to perform the mining process. In these cases, a data pre-processing task is required, transforming data into a set of new attributes, derived from the main features. In the case of RS based on explicit ratings like Collaborative Filtering (CF), the systems typically record reviews that users give to certain items, thus creating a user-item matrix in which each user and each item is associated with a row and a column, respectively. Each cell of this matrix corresponds to the rating of the user-item pair. A common transformation on the data is performed to change the grading scale, e.g. from multiple values (1-5) to a binary scale. This kind

CAMRa2010, September 30, 2010, Barcelona, Spain.

of transformations is consequence to the requirements of certain methods, like for example the simple Bayesian model of CF [9].

A typical circumstance in RS research is the impossibility that a user rates all the items. As a consequence there are many cells of the user-item matrix not defined. Various proposals have been considered for allocation of values [11], including the use of imputation techniques such as mean imputation, regression, predictive mean adjustment [12] and Bayesian multiple imputation [13]. There exists even a proposal for imputation-driven CF framework [14].

In the case of implicit CF the information available correspond to records of users activities (e.g. purchase or download a product). In such cases it is necessary to include some data processing which tend to be binary (presence or absence of an activity, although they could be of other types: e.g. to consider the recorded time observing an object in a catalog). In this way, Lee et al. [15] transform temporal information about purchasing items on pseudo-valuations. Celma [16] and Baltrunas & Amatriain [17] use information on how frequent each music track is for each user, in order to create explicit values on which it is possible to use CF algorithms.

Finally, RS including social relations are being under consideration in the last years. In this way, many different technologies based on data mining, machine learning, artificial intelligence, etc. converges in this context to better explore the outstanding relations among the systems' users. Up to date, several studies focus in the importance of dataset collection. In this sense, an initiative like the one carried out within the CAMRa Workshop [1], should be congratulated. Commonly, algorithms dealing with social relations try to explore the neighborhoods of an active user. Different techniques have been used for helping the development of recommendation systems. For example, Shepitsen [18] employ a personalization algorithm for recommendation in folksonomies which relies on hierarchical tag clusters. Other works focuses in random walk techniques, being used in many different areas and successfully brought to social recommendation as [19]. Current questions like the data sparsity or the poor prediction accuracy problems are treated by means of a factor analysis approach based on probabilistic matrix factorization, employing both users' social network information and rating records [20]. These examples, among others, emphasize the importance of research in systems dealing with social information.

2.2 Random Walks

A Markov chain is a stochastic process that links states in a graph with certain probabilities, having the property that the next state depends only on the current state, but no others. The trajectory described in a graph is called a random walk, commonly RW, represented by means of the probability of going from one node to another. The matrix representing the probabilities of going from one state to another is called transition probability matrix. The idea behind a RW applied to social graphs is that those users who have seen the same movies probably have similar taste and will be connected by, comparatively, a large number of short paths than in other case. In cases of users with different taste, the number of paths will be fewer and these paths will be longer. Here short and large are concepts related with the weight assigned to the edges connecting users in the adjacency matrix of the graph representing the process being described.

Once the transition probability matrix P has been defined, it is possible to compute the recommendations by means of different schemes:

i. Random Walk (RW)

ii. Random Walk with Restart (RWR)

All the models are based in the idea of a graph representing nodes with users and movies, interconnected by means of weighted edges. A random walk executed over the graph assumes that, starting from a node, $i = s(t)$ in time t, the random walk links nodes in terms of the transition probability matrix, P.

To compute a RW it is necessary to define the adjacency matrix of the graph. Once the symmetric adjacency matrix G is defined, we can compute the transition probability matrix P, which is a squared matrix, where each element $p_{i,j}$ represents the probability of going from node $i = s(t)$ to the adjacent node $j = s(t+1)$ as defined by equation (1):

$$p_{i,j} = P(s(t+1) = j | s(t) = i) = \frac{s_{i,j}}{\sum_{k=1}^{n} s_{i,k}} \quad (1)$$

In this equation the values of each edge in the graph are represented by means of the $s_{i,j}$ terms. We define the state transition probability matrix associated to the RW as in (2):

$$\pi(t) = [\pi_1(t), \pi_2(t), \cdots, \pi_n(t)]^T \quad (2)$$

Where $\pi_i(t) = P(s(t) = i)$ represents the probability of being at state i at time t. $\pi(t)$ represents the state probability distribution matrix at time t of the RW. The RW evolves characterized by the expressions in (3).

$$\begin{aligned} \pi(t+1) &= P^T \pi(t) \\ \pi(0) &= \pi^0 \end{aligned} \quad (3)$$

being π^0 set as the identity matrix.

In the case of RWR, is quite similar but introduces a slight different way depending on how the transition probabilities are computed. In this case, the expression to get this probability is computed as in (4):

$$\pi_i(t) = (1-a)P^T \pi_i(t) + a q_i \quad (4)$$

Starting from the node i; a represents the probability to restart at the same state i. q is a vector with zeros in all their coordinates except in the i-th position. As higher a is, as much probability to restart at the starting node. In [19] it is shown that the performance of the recommendation system using the RWR method is improved, when using the extra knowledge provided by the users' social activity.

3. EXPERIMENTS

Filmtipset is the Sweden's largest social movie recommendation community, with more than 80.000 users. It contains many social relationships that could be used to discover important information for making predictions of users' tastes. There are quite different user characteristics and relations registered. Thus, for example, genre, people in movie, film lists, film collections, etc. Using different features registered on the social network in conjunction with the implicit or explicit records of users' activities allows, by using data mining techniques, to extract new information that may be useful in the process of recommendation. Thus, for example, in the one hand we have explicit friendship relations between users

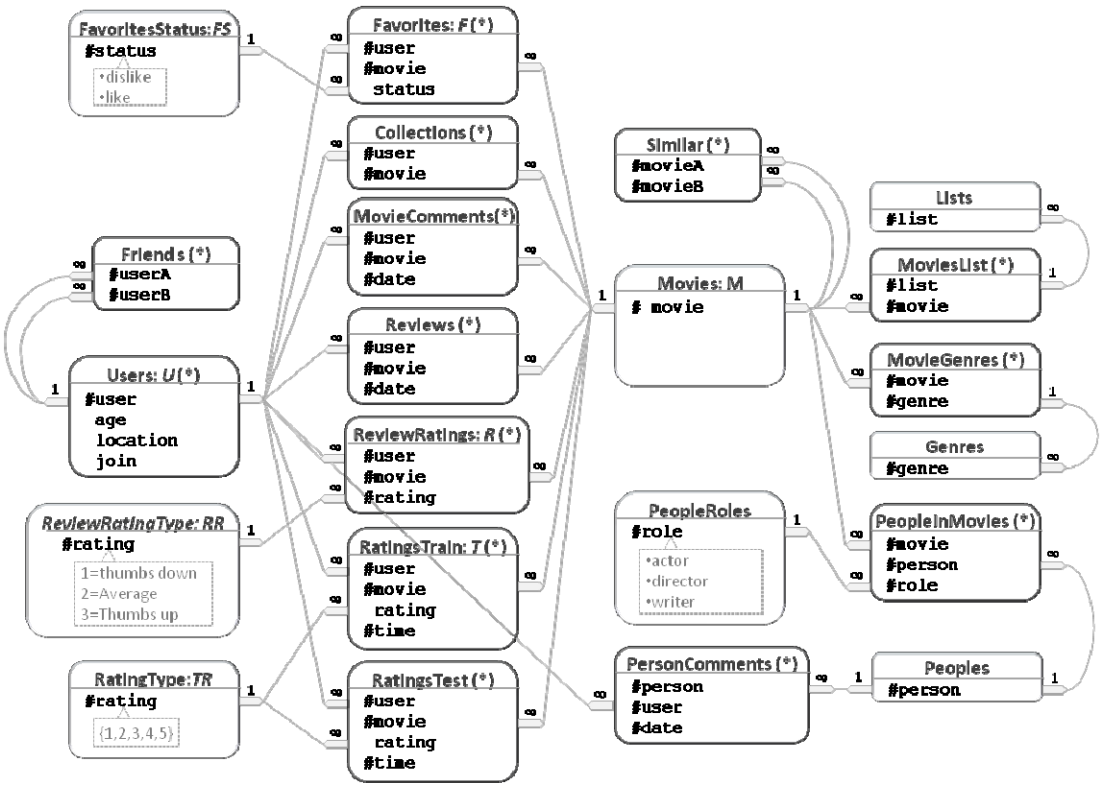

Figure 1. The model entity/relationship of the CAMRa's 2010 Filmtipset Social Recommendation dataset. Tables with (*) are those provided with the dataset.

as the ones described in the file `friends.tsv`. On the other hand, the dataset also contains implicit relations as the ones being extracted from the frequencies in which a user see a film of a specific genre or make reviews of a specific person (actor, director or writer) in different movies. This implicit information must be extracted from the tables being part of the dataset by mains of conventional database operations.

Once we have the data properly pre-processed, different algorithms can be applied to obtain the recommendations. Among the most frequently applied techniques, can be highlighted those based on the detection of nearest neighbors and those based on matrix factorization, whose popularity comes from the good results in terms of accuracy they have shown [21]. In the case of social recommendation for the social track, we have chosen an algorithm based on random walks [19], which is described in the next section.

For our experiments we have chosen the method of random walk and his variation, random walk with restart, so as to implement a recommender based on social information. We also introduced a new proposal of the method based on the personalization of the calculation of the transition probabilities in case of the restarting method, as we will explain later in section 3.3. In every case, we conducted various tests with the different implementations of the algorithm. We have got higher performance without restarts than in variants with it, as we will show later in section 4.

In the following section we describe the way by means the information needed for the definition of the adjacency matrix, consisting of several existing features in the dataset, is defined.

3.1 *Filmtipset* dataset

Figure 1 shows the model entity/relationship of the *Filmtipset* dataset. The tables distinguished by (*) are the ones provided with the dataset. Additionally, we have included the following tables *FavoritesStatus*, *ReviewRatingType*, *RatingType*, *Movies*, *PeopleRoles*, *Peoples*, *Lists* and *Genres*, to generate a database with referential integrity to optimize queries for the computations done.

It should be stressed that, to create the table *Movies*, it was taken into account existing relationship with all those tables of the *Filmtipset* dataset containing the movie, as shown in Figure 1. That is, this includes all tables except *Friends*, *Users* and *PersonComments*. We proceed in the same way to create the tables *Peoples*, *Genres* and *Lists*.

In connection with the new tables included in the database, there have been defined the following values, as shown in Figure 1:

- *FavoritesStatus.*
  $FS = \{dislike\ (0), like\ (1)\}$
- *ReviewRatingType*
  $RR = \{thumbs\ down(1), average(2), thumbs\ up(3)\}$
- *RatingType*
  $TR = \{1, 2, 3, 4, 5\}$
- *PeopleRoles*
  $PR = \{actor, director, writer\}$

In set-terms, a movie m_j of the *Filmtipset* dataset is located on the *Movies* table (M), and it is denoted $m_j \in M$. Similarly we can

express that a user u_i of the *Filmtipset* dataset, is in the *Users* table (*U*), and it is denoted $u_i \in U$.

We have focused our interest on analyzing the tables *Favorites* (*F*), *ReviewRating* (*R*) and *RatingsTest* (*T*), chosen from among those that relate users and movies. Thus for every film m_j we can establish the following average scores in each of them:

- The average favorite score of a movie m_j is computed adding all the 1's of that movie in the *Favorites* table over the total of scores of such movie. Namely:

$$f_j = \frac{1}{|U_f|} \sum_{u_k \in U_f} f_{k,j} \quad (5)$$

being U_f the set of users who scored the movie in the *Favorites* table, and $f_{k,j} \in FS$ is the score of the u_k user given to the m_j movie.

- The average review score of a movie m_j is computed adding all the scores of that movie in the *ReviewRatings* table over the total of scores of such movie. Namely:

$$r_j = \frac{1}{|U_r|} \sum_{u_k \in U_r} r_{k,j} \quad (6)$$

being U_r the set of users who scored the movie in the *ReviewRatings* table, and $r_{k,j} \in RR$ is the score of the u_k user given to the m_j movie.

- The average training score of a movie m_j is computed adding all the scores of that movie in the *RaitingsTrain* table over the total of scores of such movie. Namely:

$$t_j = \frac{1}{|U_t|} \sum_{u_k \in U_t} t_{k,j} \quad (7)$$

being U_t the set of users who scored the movie in the *RatingsTrain* table, and $t_{k,j} \in TR$ is the score of the u_k user given to the m_j movie.

3.2 User Symmetric Adjacency Matrix

3.2.1 Previous definitions

In order to implement the random walk method is necessary to define an adjacency matrix *G*, symmetric, for each user $u_a \in U$, as shown in Figure 2. It is important to emphasize this fact, that this matrix is defined for each user $u_a \in U$. Being defined this matrix, and by means of the random walk method, it is determined the list of recommended movies for that user u_a, in descending order from the scores of the row $s_{a_1}, \ldots, s_{m_n}$ in the sub-array *AS*.

In this sub-section we are going to define some concepts, criteria and sets to implement the adjacency symmetric matrix *G*.

We define *A* as the set being formed by the user u_a and his/her friends list, as defined in *Friends* table. For its part, the set *S* of selected movies represents those movies that the user u_a have not seen.

These movies are selected according to criteria described below.

- Favorite criteria: order all possible movies by average favorite score f_j.

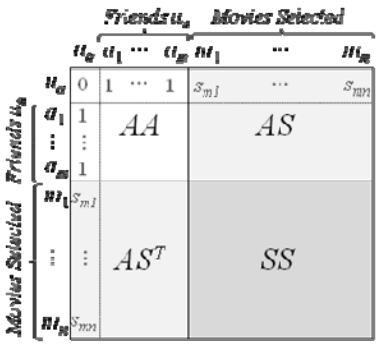

Figure 2. The social graph and their sub-matrices.

- Review criteria: order all possible movies by average review score r_j.
- Training criteria: order all possible movies by average training score t_j.
- User's friends criteria: order all the films have been seen by the u_a user's friends through the sums of scores $f_j + r_j + t_j$.
- User's similar films criteria: order all the similar films have been seen by the u_a user through the sums of scores $f_j + r_j + t_j$.
- User's movie genre criteria: the u_a user genre is the percentage of films seen on the total recorded movies with that genre. From the u_a user's genre list, the movies of the first genre are sorted by the sum $f_j + r_j + t_j$, then continues with the second genre, and so on.
- User comments to films criteria: all chosen films are those whose actors, directors or writers have been commented by the user (i.e., through the union of the tables: *PersonComments*, *People* and *PeopleInMovies*). This selection is ordered, as in previous cases, as the sum $f_j + r_j + t_j$.

For each sorting criteria the selected movies are sorted in reverse order (according to the measure used in each one). The movies that the u_a user has seen are deleted, considering the top N from the remaining ones. Once the set of movies to recommend is defined, the next step consists in scoring. N points are awarded to the first, N-1 to the second and so on. Once all the criteria have been applied to each movie in the set of movies to recommend, grades are added together and are sorted from highest to lowest score by taking the top N as the set *S*. In our case we have considered N = 1000, as standard value used by trec_eval for the computation of standard measures.

3.2.2 Definition of the graph matrix G

To construct the sub-matrix *AA* of Figure 2, we get $u_i, u_j \in A$, if u_i is a friend of u_j. In this case, the corresponding element of sub-matrix is defined $u_{ij} = u_{ji} = 1$. Otherwise, $u_{ij} = u_{ji} = 0$. The elements in the diagonal are defined $u_{ii} = 0$, considering that a user can not be friend with him/herself. Note that the matrix constructed in this way is symmetric. Moreover, by definition of the set A, $u_{aj} = u_{ja} = 1$.

Similarly, the *SS* sub-matrix is constructed, where a movie is similar to another if related through the *Similar* table.

Finally, to construct the *AS* sub-matrix, for each user $u_i \in A$, a set $S_i \subseteq S$ is defined, where S_i represent the movies m_j seen by the u_i user. We define the s_{ij} element of the *AS* sub-matrix by means of the following function:

$$s_{ij} = s_{ij}(l,\alpha,\beta,\gamma) = \begin{cases} v_j(l) & m_j \in S_i \\ w_j(\alpha,\beta,\gamma) & m_j \notin S_i \end{cases} \quad (8)$$

being $l \in \{F,R,T\}$ and $\alpha,\beta,\gamma \in \{0,1\}$. A total of twenty four combinations to implement are able with these four parameters.

For its part, the functions $v_j(l)$ and $w_j(\alpha,\beta,\gamma)$ are defined by means of the equations (9) and (10) as follows:

$$v_j(l) = \begin{cases} f_j & l = F \\ r_j & l = R \\ t_j & l = T \end{cases} \quad (9)$$

and

$$w_j(\alpha,\beta,\gamma) = \frac{\alpha f_j + \beta r_j + \gamma t_j}{max\{FS\} + max\{RR\} + max\{TR\}} \quad (10)$$

Putting *AA*, *AS* and her transposed one, and *SS* altogether, we finally get the adjacency symmetric matrix of the social graph *G* shown in Figure 2.

In section 4 the results of the application of different recommendation algorithms are presented. In Table 1, the parameters used for each method are set. For example, when specifying R-1-0-1, represents the expression

$$s_{ij} = s_{ij}(R,1,0,1) = \begin{cases} v_j(R) & m_j \in S_i \\ w_j(1,0,1) & m_j \notin S_i \end{cases} \quad (11)$$

Having into account the equations (9) and (10) jointly with (5), (6) and (7) we get that (9) become

$$v_j(R) = \frac{1}{|U_r|} \sum_{u_k \in U_r} r_{k,j}$$

and (10)

$$w_j(1,0,1) = \frac{\frac{1}{|U_f|}\sum_{u_k \in U_f} f_{k,j} + 0 + \frac{1}{|U_t|}\sum_{u_k \in U_t} t_{k,j}}{max\{FS\} + max\{RR\} + max\{TR\}}$$

3.3 Personalized RWR

Apart from these two algorithms, which are the basic ones as we can find in [19] and [22], we propose a modification in how the vector q is computed. In this case, which we call Personalized Random Walk with Restart (PRWR), each component of q in the expression (4) is computed by means of the following expression:

$$q_{ik} = \frac{1}{\sum_{s=1}^{N} R_{i,s}}; R_{i,s} = \begin{cases} 1, & p_{is} > 0 \\ 0, & p_{is} = 0 \end{cases} \quad (11)$$

For example, let $m_j \in S$ and let $u_1, \cdots, u_D$ be a set of users. Using the expression (4) as in the RWR proposal, the corresponding vector q would be $q = [0,\ldots,0,1,0,\ldots,0]$ with 1's only in the *j*-th position. Using the Personalized proposal as defined in (11), and under the hypothesis that *D'* of the *D* users have seen the movie, the resulting vector would be

$$q = \left[\frac{1}{D'}, \cdots, 0, \frac{1}{D'}, \frac{1}{D'}, \cdots, 0\right]$$

That is, we are introducing the probability to restart at state *x* as a weight for each user u_i (the same for all of them) taking into account their relative importance in the users set. The assigned value equals the restart probability to all the users who have seen the movie, not only for the one being evaluated.

We have to note that the results obtained after the execution of Random Walks, whatever the model, produce multiple ties in the values given by the recommender. This problem has a bearing on the results of precision, because they are dependent on the position occupied by the recommended items. To avoid this problem, we have ordered those films with the score tied making requests to the dataset. These requests compute the sum of the averaged scores of favorites, reviews and ratings between the *n* films tied. Finally, they are ranked using the average value.

We also tried to improve the results by redefining the values of the adjacency matrix, in order to avoid the tied recommended scores drawback. In this sense, we define a function which computes the value of s_{ij} for the *AA* sub-matrix as in (12):

$$s_{ij} = \frac{|S_i \cap S_j|}{max\{|S_i|, |S_j|\}} \quad (12)$$

Results derived from this new definition were similar to the previous ones, considering 0/1 values. This could be because of strong effect that the other sub-matrices, user-movies and movies-movies, have over the friendship relations.

In addition, we also want to report about the scores computed as ratings for the recommended movies. In our case we are not interested in rating prediction. Otherwise, we are interested in outperforming as much as possible the baseline precision. This is done by means of an utility function computing the best value, in terms of probabilities, of the transition probability matrix associated to the Random Walk.

As baselines, we have considered the work done by Liu and Lee [23], in which they compare correlation-based algorithms against social-based, where friends are used instead of nearest neighbors. Similarly, we have implemented three recommenders: the first one is a simple kNN algorithm, using Pearson correlation, the second one uses all friends as neighbors, and the last one is a combination of both, where the friends are always used and complemented with neighbors until we reach k nearest neighbors. We have to note that the best results have been found for k=15, and, because of that, we compare our algorithms against these results.

4. RESULTS

The results obtained from the experiments carried out showed us the difficulty in achieving improvements in the performance measures considered. In particular, we focused on the following measures: MAP, P@5, P@10, AUC (all suggested by the organizers), and NDGC.

We have included NDGC mainly because, in terms of Information Retrieval, if we consider users as queries, recommended items as documents resulting from the query, and the predicted ratings as approximations to the scores given by the search engine, we can compute the cumulative gain (CG) at position *p* of a particular ranking. Each user (i.e., a query) has a discounted CG. If we normalize it using the information of the whole set of users we compute the NDGC measure, which allows fair comparisons between different algorithms. Besides that, it helped us to decide which algorithm and which parameter combination (among all the

combinations tried) performed better in case of equal performances using other measures.

To compute the measures listed above we used trec_eval[1], a public program to evaluate TREC results using the standard NIST evaluation procedures. We use AUCCalculator utility [24] to calculate AUC.

In Table 1 is presented a a sample summary of some of the results obtained by the experiments, of the twenty four possible without considering the restarting probabilities, with the best performance in each case. It is worth to note that parameters column is given by the sequence *rr*-*l*-α-β-γ, where the restart rate is given by *rr* (only suitable for the algorithms with restarting capabilities, namely RWR and PRWR) and *l*-α-β-γ are the parameters for the equation (8). We underlined the parameters of the best results for each model. The best result of the whole is in bold.

The first obvious conclusion observed is that, results obtained with the RW method outperforms the ones obtained with the RWR models. In these cases, the values 0.3 and 0.5 represent the restarting probability in the RWR methods.

Secondly, all the models performed better making use of the table *ReviewRatings* feature of the dataset (that is, the *R* parameter). We conducted several trials with different parameters chosen, but always the cases with *R* gave better results than with other parameter selections. This fact is quite remarkable and unexpected, giving the idea that the information related with the reviews given by the users of the system produces better recommendations than the friendship or the ratings features.

Table 1. Summary of the results. Baselines in italics. In bold, the best result. See Section 3 for the meaning of the parameters.

Model	Parameters	MAP	P@5	P@10	AUC	NDGC
kNN	*k=15*	*0,0331*	*0,0460*	*0,0444*	*0,4325*	*0,2727*
fr		*0,0480*	*0,0524*	*0,0572*	*0,4179*	*0,3051*
fr+kNN	*k=15*	*0,0435*	*0,0282*	*0,0305*	*0,4105*	*0,3000*
RW	F-1-1-1	0,0560	0,0756	0,0690	0,4346	0,3567
	R-1-1-1	**0,0596**	**0,0802**	**0,0704**	**0,4276**	**0,3613**
	T-1-1-1	0,0531	0,0743	0,0677	0,4347	0,3530
RWR	0.3-F-1-1-1	0,0199	0,0228	0,0219	0,1678	0,2796
	0.3-R-0-1-0	0,0281	0,0337	0,0296	0,3547	0,3058
	0.3-R-1-1-1	0,0546	0,0629	0,0563	0,3922	0,3418
	0.5-R-1-1-1	0,0534	0,0574	0,0556	0,3890	0,3400
PRWR	0.3-F-1-1-1	0,0288	0,0164	0,0180	0,3075	0,3082
	0.3-R-1-1-1	0,0564	0,0433	0,0437	0,3994	0,3515
	0.5-R-1-1-1	0,0563	0,0428	0,0435	0,3986	0,3513
	0.5-T-1-1-1	0,0276	0,0009	0,0023	0,2824	0,3042

In Figure 3 it is shown the summary of MAP, P@5, P@10 and AUC using RW, RWR and PRWR methods. Whatever the method used, the best results happen in cases where all three tables of information, namely *Favorites*, *RatingsTrain* and *ReviewRatings*, are used for the recommendation. As can be seen the RW method performance is quite constant except for those cases with R-0-0-0 and T-0-0-0 parameters which presents local minimum. In this case the F parameter performs better than the *R* or the *T* ones. However, in the RWR and PRWR methods, the behavior are notably improved when the R-1-1-1 parameter is used, presenting local maxima in those cases.

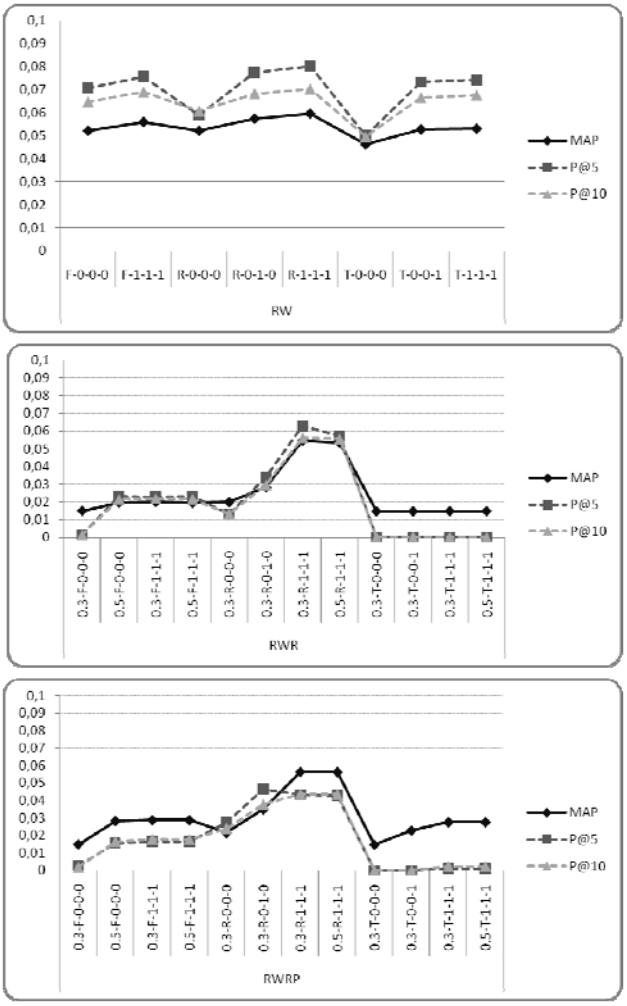

Figure 3. Summary of the MAP, P@5 and P@10 results for the RW, RWR y PRWR methods.

In Figure 4, curves at different precisions are shown, all of them using the *R* and with 1-1-1 parameters and varying, where possible, the restarting probability. It is remarkable the behavior of the PRWR model, which increase his precision from 0.0428 up to 0.681 (at P@50), whatever the restarting probability. This fact makes ourselves thought if this behavior could be transferred to lower precision values. The other restarting method, RWR, have a constant behavior. Meanwhile RW decrease their slope monotonically. At P@50, RW, RWR and PRWR are comparable. The three methods give us the idea of convergence at high precision values. It is worth interesting to note that the PRWR method has an increasing behavior.

[1] http://trec.nist.gov/trec_eval/

Finally, as we can see, every model also performs better for the values s_{ij} of the adjacency matrix computed when including all the features (that is, the 1-1-1 parameter, which means combined features). Surprisingly, the results derived from the models including restarting capabilities perform worse than the one without restarts. This effect is probably due to the way we compute the values of the adjacency matrix. The RW model is quite simple, avoiding complex data pre-processing. As we pointed above, we tried to improve the results using eq. (12) to redefine the values of s_{ij} for the *AA* sub-matrix. As could be seen, some of them (MAP and NDGC) are in the range of the best ones for each model, but none of them outperforms the previous results. In the case of AUC, the best values come from the RW method and are over the range of the baseline ones.

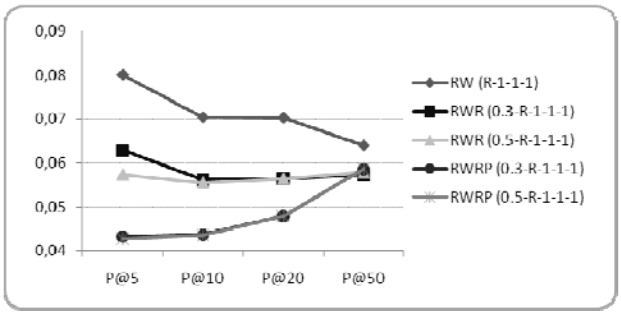

Figure 4. Precision curves at different points for RW methods with & without restart features.

5. CONCLUSIONS

In this paper, we faced CAMRa challenge using random walks as Konstas described in [19]. We have tried to adapt some of the main ideas of the model to the particular dataset delivered for the competition. The dataset is extremely rich in training data and in many types of implicit or explicit relationships, which we have not been able to exploit to the extent that we would have liked due to the lack of time. In our case we tried a slightly different method to compute de values of the adjacency matrix of the social graph, including information from different fonts (friendship relations, reviews, ratings, etc.). We also try another improved method, which we called *personalized*. Our proposal modifies the weighting scheme for the computation of transition probabilities, introducing the information of user's friends who have also seen the movies considered for the recommendation of new ones.

The greatest difficulty we have encountered stems from the need to improve outcomes in the dark, in the absence of previous values of the requested measures. We used as a baseline different types of recommenders described in the literature, which do not provide better results than those we obtained with our social recommender based in RW. There are many improvements that can be implemented. For example those including hybrid algorithms using the ones with better overall performance.

Future work requires further tests. On the one hand we want to maintain the current approach using larger adjacency matrices incorporating new features like, for example, lists of films, movie genres or people involved in movies. On the other hand we think that the use of hybrid algorithms, as mentioned above, may provide better results. In this sense we are considering to include temporal aspects. We also consider including changes in the way in which recommendation values are computed, using the techniques described by Fouss in [22] on the Laplacian matrix L+, in order to improve average computing times.

6. ACKNOWLEDGMENTS

This research was supported by the Spanish Ministry of Science and Innovation (TIN2008-06566-C04-02) and the Scientific Computing Institute at UAM. The third author also wants to acknowledge support from the Chilean Government through the Becas-Chile scholarship program.

7. REFERENCES

[1] Said, A., Berkovsky, S., De Luca, E. W. Putting Things in Context: Challenge on Context-Aware Movie Recommendation. *CAMRa2010: Proceedings of the RecSys '10 Challenge on Context-aware Movie Recommendation*, 2010

[2] Takács, G.; Pilászy, I.; Németh, B.; Tikk, D. 2009, Scalable Collaborative Filtering Approaches for Large Recommender Systems, *Journal of Machine Learning Research* **10**: 623–656

[3] L. Ramaswamy, P. Deepak, R. Polavarapu, K. Gunasekera, D. Garg, K. Visweswariah, S. Kalyanaraman. CAESAR: A Context-Aware, Social Recommender System for Low-End Mobile Devices. *Tenth International Conference on Mobile Data Management Systems, Services and Middleware*, 2009.

[4] X. Amatriain, J. M. Pujol, N. Tintarev, N. Oliver. Rate it Again: Increasing Recommendation Accuracy by User re-Rating. *Proceedings of the 2009 ACM conference on Recommender Systems*, 2009.

[5] N. Lathia, S. Hailes, L. Capra. kNN CF: A Temporal Social Network. *Proceedings of the 2008 ACM conference on Recommender systems*, 2008.

[6] X. Amatriain, N. Lathia, J. M. Pujol, H. Kwak, N. Oliver. The Wisdom of the few: a collaborative approach based on expert opinions from the web, *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 532-539.

[7] N. Lathia, S. Hailes, L. Capra. Temporal Collaborative Filtering With Adaptive Neighbourhoods. *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 796-797.

[8] T. Q. Lee, Y. Park, Y.-T.Park. An empirical study on effectiveness of temporal information as implicit ratings. *Expert Systems with Applications* 36:1315–1321, 2009.

[9] Y. Koren. Collaborative Filtering with Temporal Dynamics. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, Paris, France, 2009, pp. 447-456.

[10] K. Miyahara and M. J. Pazzani, Improvement of collaborative filtering with the simple Bayesian classifier, *Information Processing Society of Japan*, vol. 43, no. 11, 2002.

[11] X. Su, T. M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* Volume 2009, Article N. 4, 2009.

[12] R. J. A. Little, Missing-data adjustments in large surveys, *Journal of Business & Economic Statistics*, vol. 6, no. 3, pp. 287–296, 1988.

[13] D. B. Rubin, *Multiple Imputation for Nonresponse in Surveys*, John Wiley & Sons, New York, NY, USA, 1987.

[14] X. Su, T. M. Khoshgoftaar, and R. Greiner, A mixture imputation-boosted collaborative filter, *in Proceedings of the 21th International Florida Artificial Intelligence Research Society Conference* (FLAIRS '08), pp. 312–317, Coconut Grove, Fla, USA, May 2008.

[15] T. Q. Lee, Y. Park, Y.T.Park. An empirical study on effectiveness of temporal information as implicit ratings. *Expert Systems with Applications* 36:1315–1321, 2009.

[16] O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

[17] L. Baltrunas, X. Amatriain, Towards Time-Dependant Recommendation based on Implicit Feedback. In *Proceedings of the third ACM conference on Recommender systems*, New York, USA, 2009, pp. 423-424.

[18] Shepitsen, A., Gemmell, J., Mobasher, B., and Burke, R. 2008. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM Conference on Recommender Systems* (Lausanne, Switzerland, October 23 - 25, 2008). RecSys '08. ACM, New York, NY, 259-266. DOI= http://doi.acm.org/10.1145/1454008.1454048

[19] I. Konstas, V. Stathopoulos, J. M. Jose. On Social Networks and Collaborative Recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, Boston, MA, USA, pp. 195-202. 2009.

[20] Ma, H., Yang, H., Lyu, M. R., and King, I. 2008. SoRec: social recommendation using probabilistic matrix factorization. In *Proceeding of the 17th ACM Conference on information and Knowledge Management* (Napa Valley, California, USA, October 26 - 30, 2008). CIKM '08. ACM, New York, NY, 931-940. DOI= http://doi.acm.org/10.1145/1458082.1458205

[21] Bell, R. M., Bennett, J., Koren, Y., and Volinsky, C. 2009. The million dollar programming prize. *IEEE Spectr.* 46, 5 (May. 2009), 28-33. DOI= http://dx.doi.org/10.1109/MSPEC.2009.4907383

[22] Fouss, F., Pirote, A. *Randon-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation*. IEEE Trans. Knowl. Data Eng., 19(3):355-369, 2007.

[23] Liu, F. and Lee, H.J. *Use of social network information to enhance collaborative filtering performance*. Expert Systems with Appplications, 37(7):4772-4778, 2010.

[24] Davis, J. and Goadrich, M. 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international Conference on Machine Learning* (Pittsburgh, Pennsylvania, June 25 - 29, 2006). ICML '06, vol. 148. ACM, New York, NY, 233-240. DOI= http://doi.acm.org/10.1145/1143844.1143874