WEB RECOMMENDATION SYSTEM BASED ON A MARKOV-CHAIN MODEL

Francois Fouss, Stephane Faulkner, Manuel Kolp, Alain Pirotte, Marco Saerens
Information Systems Research Unit
IAG, Universite catholique de Louvain, Place des Doyens 1, B-1348 Louvain-la-Neuve, Belgium

Information Systems Research Unit
Department of Management Science, Universite of Namur, Rempart de la Vierge 8, B-5000 Namur, Belgium

Keywords: Collaborative Filtering, Markov Chains, Multi Agent System.

Abstract: This work presents some general procedures for computing dissimilarities between nodes of a weighted, undirected, graph. It is based on a Markov-chain model of random walk through the graph. This method is applied on the architecture of a Multi Agent System (MAS), in which each agent can be considered as a node and each interaction between two agents as a link. The model assigns transition probabilities to the links between agents, so that a random walker can jump from agent to agent. A quantity, called the **average first-passage time**, computes the average number of steps needed by a random walker for reaching agent k for the first time, when starting from agent i. A closely related quantity, called the **average commute time**, provides a distance measure between any pair of agents. Yet another quantity of interest, closely related to the average commute time, is the **pseudoinverse of the Laplacian matrix** of the graph, which represents a similarity measure between the nodes of the graph. These quantities, representing dissimilarities (similarities) between any two agents, have the nice property of decreasing (increasing) when the number of paths connecting two agents increases and when the "length" of any path decreases. The model is applied on a collaborative filtering task where suggestions are made about which movies people should watch based upon what they watched in the past. For the experiments, we build a MAS architecture and we instantiated the agents belief-set from a real movie database. Experimental results show that the Laplacian-pseudoinverse based similarity outperforms all the other methods.

1 INTRODUCTION

Gathering product information from large electronic catalogue on E-Commerce sites can be a time-consuming and information-overloading process. As information becomes more and more available on the World Wide Web, it becomes increasingly difficult for users to find the desired product from the millions of products available. Recommender systems have emerged in response to these issues (Breese et al., 1998), (Resnick et al., 1994), or (Shardanand and Maes, 1995). They use the opinions of members of a community to help individuals in that community to identify the information or products most likely to be interesting to them or relevant to their needs. As so, recommender systems can help E-commerce in converting web surfers into buyers by personalization of the web interface. They can also improve cross-sales by suggesting other products in which the consumer might be interested. In a world where an E-commerce site competitors are only two clicks away, gaining consumer loyalty is an essential business strategy. In this way, recommender systems can improve loyalty by creating a value-added relationship between supplier and consumer.

One of the most successful technologies for recommender systems, called collaborative filtering (CF), has been developed and improved over the past decade. For example, the GroupLens Research system (Konstan et al., 1997) provides a pseudonymous CF application for Usenet news and movies. Ringo (Shardanand and Maes, 1995) and MovieLens (Sarwar et al., 2001) are web systems that generate recommendations on music and movies respectively, suggesting collaborative filtering to be applicable to many different types of media. Moreover, some of the highest commercial web sites like Amazon.com, CDNow.com, MovieFinder.com and Launch.com made use of CF technology.

Although CF systems have been developed with success in a variety of domains, important research issues remain to be addressed in order to overcome

Fouss F., Faulkner S., Kolp M., Pirotte A. and Saerens M. (2005).
WEB RECOMMENDATION SYSTEM BASED ON A MARKOV-CHAIN MODEL.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 56-63
DOI: 10.5220/0002550700560063

two fundamental challenges: performances (e.g., the CF system can deal with a great number of consumers in a reasonable amount of time) and accuracy (e.g., users need recommendations they can trust to help them find products they will indeed like).

This paper addresses both challenges by proposing a novel method for CF. The method includes a procedure based on a Markov-chain model used for computing dissimilarities between nodes of an undirected graph. This procedure is applied on the architecture of a Multi Agent System (MAS), in which each agent can be considered as a node and each interaction among agents as a link. Moreover, MAS architectures are gaining popularity over classic ones to build robust and flexible CF applications (Wooldridge and Jennings, 1994) by distributing responsabilities among autonomous and cooperating agents.

For illustration purposes, we consider in this work a simple MAS architecture which supports an E-commerce site selling DVD movies. The MAS architecture is instantiated with three sets of agents: user agent, movie agent and movie-category agent, and two kinds of interactions: between user agent and movie agent (has_watched), and between movie agent and movie-category agent (belongs_to). Then, the procedure allows to compute dissimilarities between any pair of agents:

- Computing similarities between user agents allows to cluster them into groups with similar interest about bought movies.
- Computing similarities between user agent and movie agents allows to suggest movies to buy or not to buy.
- Computing similarities between user agent and movie-category agents allows to attach a most relevant category to each user agent.

To compute the dissimilarities, we define a random-walk model through the architecture of the MAS by assigning a transition probability to each link (i.e., interaction instance). Thus, a random walker can jump from neighbouring agents and each agent therefore represents a state of the Markov model.

From the Markov-chain model, we then compute a quantity, $m(k|i)$, called the **average first-passage time** (Kemeny and Snell, 1976), which is the average number of steps needed by a random walker for reaching state k for the first time, when starting from state i. The symmetrized quantity, $n(i,j) = m(j|i)+m(i|j)$, called the **average commute time** (Gobel and Jagers, 1974), provides a distance measure between any pair of agents. The fact that this quantity is indeed a distance on a graph has been proved independently by Klein & Randic (Klein and Randic, 1993) and Gobel & Jagers (Gobel and Jagers, 1974).

These dissimilarity quantities have the nice property of decreasing when the number of paths connecting the two agents increases and when the "length" of any path decreases. In short, two agents are considered similar if there are many short paths connecting them.

To our knowledge, while being interesting alternatives to the well-known "shortest path" or "geodesic" distance on a graph (Buckley and Harary, 1990), these quantities have not been exploited in the context of collaborative filtering; with the notable exception of (White and Smyth, 2003) who, independently of our work, investigated the use of the average first-passage time as a similarity measure between nodes. The "shortest path" distance does not have the nice property of decreasing when connections between nodes are added, therefore facilitating the communication between the nodes (it does not capture the fact that strongly connected nodes are at a smaller distance than weakly connected nodes). This fact has already been recognized in the field of mathematical chemistry where there were attempts to use the "commute time" distance instead of the "shortest path" distance (Klein and Randic, 1993). Notice that there are many different ways of computing these quantities, by using pseudoinverses or iterative procedures; details are provided in a related paper.

Section 2 introduces the random-walk model -a Markov chain model. Section 3 develops our dissimilarity measures as well as the iterative formulae to compute them. Section 4 specifies our experimental methodology. Section 5 illustrates the concepts with experimental results obtained on a MAS instantiated from the MovieLens database. Section 6 is the conclusion.

2 A MARKOV-CHAIN MODEL OF MAS ARCHITECTURE

2.1 Definition of the weighted graph

A weighted graph G is associated with a MAS architecture in the following obvious way: agents correspond to nodes of the graph and each interaction between two agents is expressed as an edge connecting the corresponding nodes.

In our movie example, this means that each instantiated agent (user agent, movie agent, and movie category agent) corresponds to a node of the graph, and each `has_watched` and `belongs_to` interaction is expressed as an edge connecting the corresponding nodes.

The weight $w_{ij} > 0$ of the edge connecting node i and node j (say there are n nodes in total) should be set to some meaningful value, with the following convention: the more important the relation between

node i and node j, the larger the value of w_{ij}, and consequently the easier the communication through the edge. Notice that we require that the weights be both positive ($w_{ij} > 0$) and symmetric ($w_{ij} = w_{ji}$). The elements a_{ij} of the adjacency matrix $\mathbf{A}$ of the graph are defined in a standard way as

$$a_{ij} = \begin{cases} w_{ij} \text{ if node } i \text{ is connected to node } j \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

where $\mathbf{A}$ is symmetric. We also introduce the Laplacian matrix $\mathbf{L}$ of the graph, defined in the usual manner:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \quad (2)$$

where $\mathbf{D} = \text{diag}(a_{i.})$ with $d_{ii} = [\mathbf{D}]_{ii} = a_{i.} = \sum_{j=1}^{n} a_{ij}$ (element i, j of $\mathbf{D}$ is $[\mathbf{D}]_{ij}$).

We also suppose that the graph is connected; that is, any node can be reached from any other node of the graph. In this case, $\mathbf{L}$ has rank $n-1$, where n is the number of nodes (Chung, 1997). If $\mathbf{e}$ is a column vector made of 1 (i.e., $\mathbf{e} = [1, 1, \ldots, 1]^{\mathrm{T}}$, where T denotes the matrix transpose) and $\mathbf{0}$ is a column vector made of 0, $\mathbf{Le} = \mathbf{0}$ and $\mathbf{e}^{\mathrm{T}}\mathbf{L} = \mathbf{0}^{\mathrm{T}}$ hold: $\mathbf{L}$ is doubly centered. The null space of $\mathbf{L}$ is therefore the one-dimensional space spanned by $\mathbf{e}$. Moreover, one can easily show that $\mathbf{L}$ is symmetric and positive semidefinite (Chung, 1997).

Because of the way the graph is defined, user agents who watch the same kind of movie, and therefore have similar taste, will have a comparatively large number of short paths connecting them. On the contrary, for user agents with different interests, we can expect that there will be fewer paths connecting them and that these paths will be longer.

2.2 A random walk model on the graph

The Markov chain describing the sequence of nodes visited by a random walker is called a random walk on a weighted graph. We associate a state of the Markov chain to every node (say n in total); we also define a random variable, $s(t)$, representing the state of the Markov model at time step t . If the random walker is in state i at time t, we say $s(t) = i$.

We define a random walk by the following single-step transition probabilities

$$\mathrm{P}(s(t+1) = j | s(t) = i) = \frac{a_{ij}}{a_{i.}} = p_{ij},$$
$$\text{where } a_{i.} = \sum_{j=1}^{n} a_{ij}$$

In other words, to any state or node i, we associate a probability of jumping to an adjacent node, $s(t+1) = j$, which is proportional to the weight w_{ij} of the edge connecting i and j. The transition probabilities only depend on the current state and not on the past ones (first-order Markov chain). Since the graph is totally connected, the Markov chain is irreducible, that is, every state can be reached from any other state. If this is not the case, the Markov chain can be decomposed into closed sets of states which are completely independent (there is no communication between them), each closed set being irreducible.

Now, if we denote the probability of being in state i at time t by $x_i(t) = \mathrm{P}(s(t) = i)$ and we define $\mathbf{P}$ as the transition matrix whose entries are $p_{ij} = \mathrm{P}(s(t+1) = j | s(t) = i)$, the evolution of the Markov chain is characterized by

$$\begin{cases} x_i(0) = x_i^0 \\ x_i(t+1) = \mathrm{P}(s(t+1) = i) \\ = \sum_{j=1}^{n} \mathrm{P}(s(t+1) = i | s(t) = j)\, x_j(t) \\ = \sum_{j=1}^{n} p_{ji}\, x_j(t) \end{cases}$$

Or, in matrix form,

$$\begin{cases} \mathbf{x}(0) = \mathbf{x}^0 \\ \mathbf{x}(t+1) = \mathbf{P}^{\mathrm{T}}\mathbf{x}(t) \end{cases} \quad (3)$$

where T is the matrix transpose.

This provides the state probability distribution $\mathbf{x}(t) = [x_1(t), x_2(t), ..., x_n(t)]^{\mathrm{T}}$ at time t once the initial probability density, $\mathbf{x}^0$, is known. For more details on Markov chains, the reader is invited to consult standard textbooks on the subject (Bremaud, 1999), (Kemeny and Snell, 1976), (Norris, 1997).

3 AVERAGE FIRST-PASSAGE TIME AND AVERAGE COMMUTE TIME

In this section, we review two basic quantities that can be computed from the definition of the Markov chain, that is, from its probability transition matrix: the average first-passage time and the average commute time. Relationships allowing to compute these quantities are derived in a heuristic way (see, e.g., (Kemeny and Snell, 1976) for a more formal treatment).

3.1 The average first-passage time

The average first-passage time, $m(k|i)$ is defined as the average number of steps a random walker, starting in state $i \neq k$, will take to enter state k for the first time (Norris, 1997). More precisely, we define the

minimum time until absorption by state k as $T_{ik} = \min(t \geq 0 \mid s(t) = k \text{ and } s(0) = i)$ for one realization of the stochastic process. The average first-passage time is the expectation of this quantity, when starting from state i: $m(k|i) = E[T_{ik}|s(0) = i]$.

We show in a related paper how to derive a recurrence relation for computing $m(k|i)$ by first-step analysis.

We obtain

$$\begin{cases} m(k|i) = 1 + \sum_{j=1; j \neq k}^{n} p_{ij}\, m(k|j), \text{ for } i \neq k \\ m(k|k) = 0 \end{cases} \quad (4)$$

These equations can be used in order to iteratively compute the first-passage times (Norris, 1997). The meaning of these formulae is quite obvious: in order to go from state i to state k, one has to go to any adjacent state j and proceed from there.

3.2 The average commute time

We now introduce a closely related quantity, the average commute time, $n(i, j)$, which is defined as the average number of steps a random walker, starting in state $i \neq j$, will take before entering a given state j for the first time, and go back to i. That is, $n(i, j) = m(j|i) + m(i|j)$. Notice that, while $n(i, j)$ is symmetric by definition, $m(i|j)$ is not.

3.3 The average commute time is a distance

As shown by several authors (Gobel and Jagers, 1974), (Klein and Randic, 1993), the average commute time is a distance measure, since, for any states i, j, k:

$$\begin{cases} n(i, j) \geq 0 \\ n(i, j) = 0 \text{ if and only if } i = j \\ n(i, j) = n(j, i) \\ n(i, j) \leq n(i, k) + n(k, j) \end{cases}$$

Another important point not proved here is that $\mathbf{L}^+$ is a matrix whose elements are the inner products of the node vectors embedded in an Euclidean space preserving the ECTD between the nodes in this Euclidean space, the node vectors are exactly separated by ECTD. $\mathbf{L}^+$ can therefore be considered as a similarity matrix between the nodes (as in the vectors space model in information retrieval).

In summary, three basic quantities will be used as providing a dissimilarity/similarity measure between nodes: the average first-passage time, the average commute time, and the pseudoinverse of the Laplacian matrix.

4 EXPERIMENTAL METHODOLOGY

Remember that each agent of the three sets corresponds to a node of the graph. Each node of the user-agent set is connected by an edge to the watched movies of the movie-agent set. In all these experiments we do not take the movie-category agent set into account in order to perform fair comparisons between the different methods. Indeed, two scoring algorithms (i.e., cosine and nearest-neighbours algorithms) cannot naturally use the movie-category set to rank the movies.

4.1 Data set

For these experiments, we developed a MAS architecture corresponding to our movie example. The belief set of the user agents, movie agents, and movie-category agents has been instantiated from the real MovieLens database (www.movielens.umn.edu). Each week hundreds of users visit MovieLens to rate and receive recommendations for movies.

We used a sample of this database proposed in (Sarwar et al., 2002). Enough users were randomly selected to obtain 100,000 ratings (considering only users that had rated 20 or more movies). The database was then divided into a `training set` and a `test set` (which contains 10 ratings for each of 943 users). The `training set` set was converted into a 2625 x 2625 matrix (943 user agents, and 1682 movie agents that were rated by at least one of the user agents). The results shown here do not take into account of the ratings provided by the user agents here (the experiments using the ratings gave similar results) but only the fact that a user agent has or has not interacted with a movie agent (i.e., the user-movie matrix is filled in with 0's or 1's).

We then applied the methods described in Section 4.2 to the `training set` and compared the results thanks to the `test set`.

4.2 Scoring algorithms

Each method supplies, for each user agent, a set of similarities (called scores) indicating preferences about the movies, as computed by the method. Technically, these scores are derived from the computation of dissimilarities between the user-agent nodes and the movie-agent nodes. The movie agents that are closest to an user agent, in terms of this dissimilarity score, (and that have not been watched) are considered the most relevant.

The first four scoring algorithms are based on the average first-passage time and are computed from

the probability transition matrix of the corresponding Markov model.

Average commute time (CT). We use the average commute time, $n(i,j)$, to rank the agents of the considered set, where i is an agent of the user-agent set and j is an agent of the set to which we compute the dissimilarity (the movie-agent set). For instance, if we want to suggest movies to people for watching, we will compute the average commute time between user agents and movie agents. The lower the value is, the more similar the two agents are. In the sequel, this quantity will simply be referred to as "commute time".

Principal components analysis defined on average commute times (PCA CT). In a related paper, we showed that, based on the eigenvector decomposition, the nodes vectors, $\mathbf{e}_i$, can be mapped into a new Euclidean space (with 2625 dimensions in this case) that preserves the Euclidean Commute Time Distance (ECTD), or a $m-$dimensional subspace keeping as much variance as possible, in terms of ECTD. We varied the dimension of the subspace, m, from 25 to 2625 by step of 25. It shows the percentage of variance accounted for by the m first principal components $\sum_{i=1}^{m}\lambda_i/\sum_{j=1}^{n}\lambda_j$. After performing a PCA and keeping a given number of principal components, we recompute the distances in this reduced subspace. These Euclidean commute time distances between user agents and movie agents are then used in order to rank the movies for each user agent (the closest first). The best results were obtained for 100 dimensions ($m = 100$).

Notice that, in the related paper, we also shows that this decomposition is similar to principal components analysis in the sense that the projection has maximal variance among all the possible candidate projections.

Average first-passage time (one-way). In a similar way, we use the average first-passage time, $m(i|j)$, to rank agent i of a the movie-agent set with respect to agent j of the user-agent set. This provides a dissimilarity between agent j and any agent i of the considered set. This quantity will simply be referred to as "one-way time".

Average first-passage time (return). As a dissimilarity between agent j of the user-agent set and agent i of the movie-agent set, we now use $m(j|i)$ (the transpose of $m(i|j)$), that is, the average time used to reach j (from the user-agent set) when starting from i. This quantity will simply be referred to as "return time".

We now introduce other standard collaborative filtering methods to which we will compare our algorithms based on first-passage time.

Nearest neighbours (KNN). The nearest neighbours method is one of the simplest and oldest methods for performing general classification tasks. It can be represented by the following rule: to classify an unknown pattern, choose the class of the nearest example in the training set as measured by a similarity metric. When choosing the k-nearest examples to classify the unknown pattern, one speaks about k-nearest neighbours techniques.

Table 1: Contingency table.

		Individual j		
		1	0	Totals
Individual i	1	a	b	$a+b$
	0	c	d	$c+d$
Totals		$a+c$	$b+d$	$p=a+b+c+d$

Using a nearest neighbours technique requires a measure of "closeness", or "similarity". There is often a great deal of subjectivity involved in the choice of a similarity measure (Johnson and Wichern, 2002). Important considerations include the nature of the variables (discrete, continuous, binary), scales of measurement (nominal, ordinal, interval, ratio), and subject matter knowledge.

In the case of our MAS movie architecture, pairs of agents are compared on the basis of the presence or absence of certain features. Similar agents have more features in common than do dissimilar agents. The presence or absence of a feature is described mathematically by using a binary variable, which assumes the value 1 if the feature is present (if the person i has watched the movie k, that is if the user agent i has an interaction with movie agent k) and the value 0 if the feature is absent (if the person i has not watched the movie k, that is if the user agent i has no interaction with movie agent k).

More precisely, each agent i is characterized by a binary vector, $\mathbf{v}_i$, encoding the interactions with the movie agents (remember that there is an interaction between an user agent and a movie agent if the considered user has watched the considered movie). The nearest neighbours of agent i are computed by taking the k nearest $\mathbf{v}_j$ according to a given similarity measure between binary vectors, $\mathrm{sim}(i,j) = \mathrm{sim}(\mathbf{v}_i,\mathbf{v}_j)$. We performed systematic comparisons between eight different such measures (see (Johnson and Wichern, 2002), p.674). Based on these comparisons, we retained the measure that provide the best results: $a/(b+c)$, where a, b, c and d are defined in Table 1. In this table, a represents the frequency of 1-1 matches between $\mathbf{v}_i$ and $\mathbf{v}_j$, b is the frequency of 1-0 matches, and so forth.

We also varied systematically the number of neighbours k $(= 10, 20, ..., 940)$. The best score was obtained with 110 neighbours.

In Section 5, we only present the results obtained by the best k-nearest neighbours model (i.e.,

$\mathrm{sim}(i,j) = a/(b+c)$ and $k = 110$).

Once the k-nearest neighbours are computed, the movie agents that are proposed to user agent i are those that have the highest predicted values. The predicted value of user agent i for movie agent j is computed as a sum weighted by sim of the values (0 or 1) of item j for the neighbours of user agent i:

$$pred(i,j) = \frac{\sum_{p=1}^{k} sim(i,p)\, a_{pj}}{\sum_{p=1}^{k} sim(i,p)} \qquad (5)$$

where a_{pj} is defined in Equation 1 and we keep only the k nearest neighbours.

Cosine coefficient. The cosine coefficient between user agents i and j , which measures the strength and the direction of a linear relationship between two variables, is defined by $\mathrm{sim}(i,j) = (\mathbf{v}_i{}^{\mathrm{T}}\mathbf{v}_j)/(\|\mathbf{v}_i\|\,\|\mathbf{v}_j\|)$.

The predicted value of user agent i for movie agent j, considering 60 neighbours (i.e., $k = 60$), is computed in a similar way as in the k-nearest neighbours method (see Equation 5).

Dunham overviews in (Dunham, 2003) other similarity measures related to cosine coefficient (i.e., Dice similarity, Jaccard similarity and Overlap similarity). In Section 5, we only show the results for the cosine coefficient, the other methods giving very close results.

Katz. This similarity index has been proposed in the social sciences field. In his attempt to find a new social status index for evaluating status in a manner free from the deficiencies of popularity contest procedures, Katz proposed in (Katz, 1953) a method of computing similarities, taking into account not only the number of direct links between items but, also, the number of indirect links (going through intermediaries) between items.

The similarity matrix is

$$\mathbf{T} = \alpha\mathbf{A} + \alpha^2\mathbf{A}^2 + ... + \alpha^k\mathbf{A}^k + ... = (\mathbf{I} - \alpha\mathbf{A})^{-1} - \mathbf{I}$$

where $\mathbf{A}$ is the adjacency matrix and α is a constant which has the force of a probability of effectiveness of a single link. A k-step chain or path, then, has probability α^k of being effective. In this sense, α actually measures the non-attenuation in a link, $\alpha = 0$ corresponding to complete attenuation and $\alpha = 1$ to absence of any attenuation. For the series to be convergent, α must be less than the inverse of the spectral radius of $\mathbf{A}$.

For the experiment, we varied systematically the value of α and we only present the results obtained by the best model (i.e., $\alpha = 0.01$ * (spectral radius)$^{-1}$).

Once we have computed the similarity matrix, the closest movie agent representing a movie that has not been watched is proposed first to the user agent.

Dijkstra's algorithm. Dijkstra's algorithm solves a shortest path problem for a directed and connected graph which has nonnegative edge weights. As a distance between two agents of theMAS architecture, we compute the shortest path between these two agents. The closest movie agent representing a movie that has not been watched is proposed first to the user agent.

Pseudoinverse of the Laplacian matrix ($\mathbf{L}^+$). The pseudoinverse of the Laplacian matrix provides a similarity measure since $\mathbf{L}^+$ is the matrix containing the inner product of the vectors in the transformed space where the nodes are exactly separated by the ECTD (details are provided in a related paper). The predicted value of user agent i for movie agent j, considering 100 neighbours (i.e., $k = 100$), is computed in a similar way as in the k-nearest neighbours method (see Equation 5).

4.3 Performance evaluation

The performances of the scoring algorithms will be assessed by a variant of Somers'D, the degree of agreement (Siegel and Castellan, 1988).

For computing this degree of agreement, we consider each possible pair of movie agents and determine if our method ranks the two agents of each pair in the correct order (in comparison with the `test set` which contains watched movies that should be ranked first) or not. The degree of agreement is therefore the proportion of pairs ranked in the correct order with respect to the total number of pairs, without considering those for which there is no preference. A degree of agreement of 0.5 (50% of all the pairs are in correct order and 50% are in bad order) is similar to a completely random ranking. On the other hand, a degree of agreement of 1 means that the proposed ranking is identical to the ideal ranking.

5 RESULTS

5.1 Ranking procedure

For each user agent, we first select the movie agents representing movies that have not been watched. Then, we rank them according to one of the proposed scoring algorithms. Finally, we compare the proposed ranking with the `test set` (if the ranking procedure performs well, we expect watched movies belonging to the `test set` to be on top of the list) by using the degree of agreement.

5.2 Results and discussions

The results of the comparison are tabulated in Table 2 (where we display the degree of agreement for each

Table 2: Results obtained by the ranking procedures without considering the movie-category set.

CT	PCA CT	One-way	Return	Katz
0.8566	0.8710	0.8564	0.8065	0.8790

KNN	Dijkstra	Cosine	$\mathbf{L}^+$
0.9266	0.5034	0.9273	0.9302

Table 3: Time (in sec) needed to compute predictions for all the non-watched movies and all the users

CT	PCA CT	One-way	Return	Katz
463.9	2173.5	464.9	466.19	66.9

KNN	Dijkstra	Cosine	$\mathbf{L}^+$
19.6	5606.1	348.46	621.9

method). We used the `test set` (which includes 10 movies for each of the 943 users) to compute the global degree of agreement.

Based on Table 2, we observe that the best degree of agreement is obtained by the $\mathbf{L}^+$ method (0.9302). The next degrees of agreement are obtained by the Cosine (0.9273) and the k-nearest neighbours method (0.9266). It is also observed that the commute time and the average first-passage time (one-way) provide good results too, but are outperformed by the Cosine, the KNN, Katz' algorithm (0.8790), and the PCA ($0,8710$). They present a degree of agreement of 0.8566 and 0.8564 respectively. Notice, however, that the results of the $\mathbf{L}^+$ method, the Cosine, the KNN, and the PCA are purely indicative, since they highly depend on the appropriate number of neighbours or on the appropriate number of principal components, which are difficult to estimate a priori. The commute time and the average first-passage time (one-way) outperform the average first-passage time (return) (0.8065). A method provides much worse results: Dijkstra's algorithm (0.5034). It seems that, for Dijkstra algorithm, nearly each movie agent can be reached from any user agent with a shortest path distance of 3. The degree of agreement is therefore close to 0.5 because of the difficulty to rank the movies agent.

5.3 Computational issues

In this section, we perform a comparison of the computing times (for a Pentium 4, 2.40 GHz) for all the implemented methods: the average commute time, the principal components analysis (we consider 10 components), the average first-passage time one-way and return, the Katz method, the k-nearest neighbours (we consider 10 neighbours), the Dijkstra algorithm, the Cosine method (we consider again 10 neighbours), and the $\mathbf{L}^+$ method. Table 3 shows the times, in seconds (using the Matlab `cputime` function), needed by each method to provide predictions for all the non-watched movies agent and for each user agent (i.e., 943 user agents).

We observe on the one hand, that the fastest method is the k-nearest neighbours method and one the other hand, that the slowest methods are PCA and Dijkstra algorithm. The method which provides the best degree of agreement (i.e., using the $\mathbf{L}^+$ matrix as similarity measure) takes much more time than the k-nearest neighbours method but is quite as fast as the Markov-based algorithms.

6 CONCLUSIONS AND FURTHER WORK

We introduced a general procedure for computing dissimilarities between agents of a MAS architecture. It is based on a particular Markov-chain model of random walk through the graph. More precisely, we compute quantities (average first-passage time, average commute time, and the pseudoinverse of the Laplacian matrix) that provide dissimilarity measures between any pair of agents in the system.

We showed through experiments performed on MAS architecture instantiated from the MovieLens database that these quantities perform well in comparison with standard methods. In fact, as already stressed by (Klein and Randic, 1993), the introduced quantities provide a very general mechanism for computing similarities between nodes of a graph, by exploiting its structure.

We are now investigating ways to improve the Markov-chain based methods.

The main drawback of these methods is that it does not scale well for large MAS. Indeed, the Markov model has as many states as agents in the MAS. Thus, in the case of large MAS, we should rely on the sparseness of the data matrix as well as on iterative formulae (such as Equation 4).

REFERENCES

Breese, J., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*.

Bremaud, P. (1999). *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer-Verlag.

Buckley, F. and Harary, F. (1990). *Distance in graphs*. Addison-Wesley Publishing Company.

Chung, F. R. (1997). *Spectral Graph Theory*. American Mathematical Society.

Dunham, M. (2003). *Data Mining: Introductory and Advanced Topics*. Prentice Hall.

Gobel, F. and Jagers, A. (1974). Random walks on graphs. *Stochastic Processes and their Applications*, 2:311–336.

Johnson, R. and Wichern, D. (2002). *Applied Multivariate Statistical Analysis, 5th Ed.* Prentice Hall.

Katz, L. (1953). A new status index derived from sociometric analysis. *Psychmetrika*, 18(1):39–43.

Kemeny, J. G. and Snell, J. L. (1976). *Finite Markov Chains*. Springer-Verlag.

Klein, D. J. and Randic, M. (1993). Resistance distance. *Journal of Mathematical Chemistry*, 12:81–95.

Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., and Riedl, J. (1997). Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87.

Norris, J. (1997). *Markov Chains*. Cambridge University Press.

Resnick, P., Neophytos, I., Mitesh, S., Bergstrom, P., and Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 175–186.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the International World Wide Web Conference*, pages 285–295.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2002). Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. *Proceedings of the Fifth International Conference on Computer and Information Technology*.

Shardanand, U. and Maes, P. (1995). Social information filtering: Algorithms for automating 'word of mouth'. *Proceedings of the Conference on Human Factors in Computing Systems*, pages 210–217.

Siegel, S. and Castellan, J. (1988). *Nonparametric Statistics for the Behavioral Sciences, 2nd Ed.* McGraw-Hill.

White, S. and Smyth, P. (2003). Algorithms for estimating relative importance in networks. *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 266–275.

Wooldridge, M. and Jennings, N. R. (1994). Intelligent agents: Theory and practice. *Knowledge Engineering Review paper*, 2:115–152.