A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation

Francois Fouss, Alain Pirotte & Marco Saerens
Information Systems Research Unit (ISYS/IAG)
Université catholique de Louvain
Place des Doyens 1
B-1348 Louvain-la-Neuve, Belgium
{fouss, pirotte, saerens}@isys.ucl.ac.be

Abstract

This work presents a new perspective on characterizing the similarity between elements of a database or, more generally, nodes of a weighted, undirected, graph. It is based on a Markov-chain model of random walk through the database. The suggested quantities, representing dissimilarities (or similarities) between any two elements, have the nice property of decreasing (increasing) when the number of paths connecting those elements increases and when the "length" of any path decreases. The model is evaluated on a collaborative recommendation task where suggestions are made about which movies people should watch based upon what they watched in the past. The model, which nicely fits into the so-called "statistical relational learning" framework as well as the "link analysis " paradigm, could also be used to compute document or word similarities, and, more generally, could be applied to other database or web mining tasks.

1 Introduction

This work views a database as a collection of element sets connected by relationships. The model exploits the graph structure of the database to compute a dissimilarity measure between elements. Computing dissimilarities between pairs of elements allows, for instance, to determine the item that is most relevant (similar) to a given item. Also, elements in a set can be assigned a category provided by elements from another set. Computing dissimilarities between elements of the same set amounts to a clustering task.

For example, imagine a simple movie database with three sets of elements (tables), `people`, `movie`, and `movie_category`, and two relationships `has_watched`, between `people` and `movie`, and `belongs_to`, between `movie` and `movie_category`.

(1) Computing similarities between people allows to cluster them into groups with similar interest about watched movies.

(2) Computing similarities between people and movies allows to suggest movies to watch or not to watch.

(3) Computing similarities between people and movie categories allows to attach a most relevant category to each person.

Notice, however, that all the developments remain valid in the more general context of computing dissimilarities between nodes of a **weighted**, **undirected**, **graph**.

The paper is structured as follows. Section 2 introduces the random-walk model (a Markov-chain model) as well as our similarity measures. Section 3 specifies our experimental methodology. Section 4 illustrates the concepts with experimental results obtained on the MovieLens database. Section 5 is the conclusion.

2 A Markov-chain model of database navigation

The procedure used to compute dissimilarities based on a Markov-chain model is described in details in [3] or [11]; we now briefly summarize the suggested model.

2.1 Definition of the weighted graph

A weighted graph G is associated with a database in the following obvious way: database elements correspond to nodes of the graph and database links correspond to edges.

In our movie example, this means that each element of the `people`, `movie` and `movie_category` sets (tables) corresponds to a node of the graph, and each `has_watched` and `belongs_to` link is expressed as an edge connecting the corresponding nodes. Because of the way the graph is defined, people who watch the same kind of movie, and therefore have similar taste, will be connected

by a comparatively large number of short paths. On the contrary, for people with different interests, we can expect that there will be fewer paths connecting them and that these paths will be longer.

The weight w_{ij} of the edge connecting node i and node j 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 the weights to be both positive ($w_{ij} > 0$) and symmetric ($w_{ij} = w_{ji}$). For instance, for an `has_watched` edge, the weight could be set to the number of times the person watched the corresponding movie. The elements a_{ij} of the adjacency matrix $\mathbf{A}$ (which is a symmetric matrix) of the graph are defined in a standard way as $a_{ij} = w_{ij}$ if node i is connected to node j and $a_{ij} = 0$ otherwise. We also introduce the Laplacian matrix $\mathbf{L}$ of the graph, defined in the usual manner: $\mathbf{L} = \mathbf{D} - \mathbf{A}$ as well as its Moore-Penrose pseudoinverse $\mathbf{L}^+$. Here, $\mathbf{D} = \mathrm{diag}(a_{i.})$ with $d_{ii} = [\mathbf{D}]_{ii} = a_{i.} = \sum_{j=1}^{n} a_{ij}$ (element i, j of $\mathbf{D}$ is $[\mathbf{D}]_{ij} = d_{ij}$), if there are n nodes in total.

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. We associate a state of the Markov chain to every node; 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, then $s(t) = i$. Thus we define a random walk with the following single-step transition probabilities $\mathrm{P}(s(t+1) = j | s(t) = i) = a_{ij}/a_{i.} = p_{ij}$, where $a_{i.} = \sum_{j=1}^{n} a_{ij}$.

In other words, to any state or node $s(t) = i$, 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). We also suppose that the graph is connected, that is, any node can be reached from any other node. Therefore, 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 should be decomposed into closed sets of states which are completely independent (there is no communication between them), each closed set being irreducible.

2.3 Average first-passage time, average commute time and the pseudoinverse of the Laplacian matrix

In this subsection, we review three basic quantities that can be computed from the definition of the Markov chain, that is, from its transition probability matrix: the average first-passage time, the average commute time, and the pseudoinverse of the Laplacian matrix.

The **average first-passage time** $m(k|i)$ is defined as the average number of steps that a random walker, starting in state $i \neq k$, will take to enter state k for the first time [10]. In [3], we show that the average first-passage time can be computed in terms of $\mathbf{L}^+$: $m(k|i) = \sum_{j=1}^{n} \left(l_{ij}^+ - l_{ik}^+ - l_{kj}^+ + l_{kk}^+ \right) d_{jj}$ where $l_{ij}^+ = [\mathbf{L}^+]_{ij}$.

The **average commute time** $n(i, j)$ is defined as the average number of steps that a random walker, starting in state $i \neq j$, will take before entering 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. It can be shown that this quantity is a distance measure on a graph; this fact was proved independently by Klein & Randic [8] and Gobel & Jagers [4]. The average commute time can also be computed in terms of $\mathbf{L}^+$: $n(i, j) = V_G \left(l_{ii}^+ + l_{jj}^+ - 2l_{ij}^+ \right)$ where V_G is the volume of the graph, $V_G = \sum_{i,j=1}^{n} a_{ij}$. The square root of this quantity, $\sqrt{n(i, j)}$, also defines a distance measure between the nodes of the graph, and is called the **Euclidean Commute Time Distance** (ECTD; see [3], [11]). This distance measure has a number of interesting properties; for instance, it is Euclidean: the nodes of the graph can be represented in an Euclidean space where each couple of nodes is exactly separated by the ECTD.

Moreover, it can be shown [3], [11] that the **pseudoinverse of the Laplacian matrix** of the graph ($\mathbf{L}^+$) contains the inner products of the node vectors in this Euclidean space completely preserving the ECTD (an Euclidean space where the nodes are exactly separated by the ECTD). It can therefore be considered as a similarity measure between nodes and it is a valid kernel (a Mercer kernel [13]).

All these quantities have the nice property of decreasing (or increasing, depending on whether the quantity is a dissimilarity or a similarity measure) when the number of paths connecting two elements increases and when the "length" of any path decreases (the communication is facilitated). In short, two elements 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 [1], those quantities have not been exploited in the context of collaborative recommendation or, more generally, in web mining or machine learning, with the notable exception of White and Smith [15] who, independently of our work, investigated the use of the average first-passage time as a similarity measure between nodes. Notice that the "shortest path" distance does not have the nice property of decreasing when connections between nodes are added (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 [8].

Notice also that this approach has a number of interesting relationships with spectral clustering or embedding, which are detailed in [11]. It also fits quite naturally into the so-called "statistical relational learning" frameworks (see for instance [2] and the other papers in this special issue) or link analysis [9]; indeed, the objective here is to mine a database by exploiting the relations (the links) between the tables.

3 Experimental methodology

Remember that each element of the `people` and the `movie` sets corresponds to a node of the graph. Each node of the `people` set is connected by an edge to the movies watched by the corresponding person. In our experiments, if not otherwise mentioned, we do not take the `movie_category` set into account in order to perform fair comparisons between the different methods. Indeed, two standard scoring algorithms (i.e., cosine and nearest-neighbours algorithms) cannot naturally use the `movie_category` set to rank the movies.

3.1 Data set description

Our experiments were performed on a real movie database from the web-based recommender system MovieLens (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 [12]. Enough users (i.e., 943 users) were randomly selected to obtain 100,000 ratings (considering only users that had rated 20 or more movies on a total of 1682 movies).

The experiments are divided into two parts. A **first experiment** is performed in order to tune the parameters of all the methods (Section 4.2: preliminary results) while the **second experiment** aims to further assess the different methods by cross-validation (Section 4.3: cross-validation results).

In the first experimental setting, the data set containing the ratings (the links of the graph) was divided into a `training set` and a `test set`, by splitting the data in the standard way, as suggested in [12]. They both concern the same sets of users and movies. The `test set` contains 10 ratings for each of the 943 users (about 10,000 ratings), while the `training set` contains the remainder of the ratings (about 90,000 ratings). No link (i.e., a rating of a particular user for a specific movie) belongs both to the `training set` and to the `test set`.

In the second experimental setting, in order to obtain a more accurate assessment of the performances, the data set is divided into n subsets ($n = 5$), and the method is applied n times (5-fold cross-validation). Each time, one of the n subsets is used as the `test set` and the other $n - 1$ subsets are put together to form a `training set`. Then the average result across all n trials is computed. In summary, there are two differences with the first experimental setting: (1) The `test set` contains 20% of the ratings (instead of about 10% in the first experimental setting); (2) For each user, the `test set` may contain $0, 1, ...10$ or more ratings (instead of 10 exactly for the first experimental setting)

In both cases, the `training set` was converted into a 2625×2625 matrix (943 users and 1682 movies rated by at least one of the users). Notice that the results shown here do not take into account the numerical value of the ratings provided by the users (the experiments using them gave similar results) but only the fact that a user has or has not watched a movie (i.e., entries in the user-movie matrix are 0's and 1's).

We then applied the methods described in Section 3.3 to the `training set` and compared the performances thanks to the `test set`.

Each method supplies, for each person, 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 (or similarities) between the people nodes and the movie nodes. From that information, we extract, for each user, a ranked list of all the movies that the user has not watched, according to the training set. In that list, the movies closest to the person, in terms of the dissimilarity score, are considered as the most relevant.

3.2 Performance evaluation

The performance of the scoring algorithms (see Section 3.3) is assessed by a variant of Somers'D, the **degree of agreement** [14].

For each user, the `test set` contains a set of movies (10 in the first experiment; an arbitrary number in the second experiment) that the user has watched but removed from the `training set` (in other words, the links to these movies are simply deleted from the graph). Those movies are part of the ranked list supplied by each method which also contains all the movies that the user has not watched, according to the `training set`.

To compute the degree of agreement, we consider each pair of movies (where one movie is in the `test set` for that user and the other movie is in the ranked list computed from the `training set` and not in the `test set`). A method ranks the pair in the correct order if, in the ranked list, the movie from the `test set` (watched movie) precedes the movie not in the `test set` (non-watched movie). The degree of agreement is the proportion of pairs ranked in the correct order with respect to the total num-

ber of pairs. The idea here is that the scoring method should predict the movies that have indeed been watched (`test set`) by ranking them first.

The results discussed in the next section provide a single degree of agreement for all the users. 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 1.0 (100%) means that the proposed ranking is identical to the ideal ranking (the watched movies of the `test set` are ranked first).

3.3 Scoring algorithms

Ten different scoring, or ranking, methods are compared. The first scoring algorithm, the maximum frequency algorithm, will be considered as the reference method. The next five scoring algorithms are based on the introduced Markov model providing a dissimilarity/similarity measure between nodes: the average commute time (normal and PCA-based), the average first-passage time (one-way and return), and the pseudoinverse of the Laplacian matrix.

These Markov-based methods will be compared to four standard techniques introduced below: (1) k-nearest neighbours techniques; (2) cosine coefficient; (3) Katz' method; (4) shortest path algorithm. We now describe these methods in more details.

Maximum frequency algorithm (MaxF). In the maximum frequency method, we trivially rank the movies by the number of times each movie has been watched by the users. In other words, the most watched movie (the best-seller) is simply suggested first to each user. The ranking is thus the same for all the users. MaxF will be considered as a reference to which all the suggested methods will be compared; it may be viewed as an equivalent of basing the decision only on the a priori probabilities in supervised classification.

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

Principal components analysis based on ECTD (PCA CT). In [11], we showed that, based on the eigenvector decomposition, the nodes can be mapped into a new Euclidean space (with more than 2600 dimensions in this case) that preserves the ECTD, or a m-dimensional subspace keeping as much variance as possible, in terms of ECTD.

Thus, after performing a PCA and keeping a given number of principal components, we recompute the distances in this reduced subspace. These approximate ECTD between people and movies are then used in order to rank the movies for each user (the closest first).We varied systematically the dimension of the subspace, m, from 10 to 2620 by step of 10. The best results were obtained for 60 principal components ($m = 60$).

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

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

Pseudoinverse of the Laplacian matrix ($\mathbf{L}^+$). $\mathbf{L}^+$ provides a similarity measure since it is the matrix containing the inner products of the node vectors in the Euclidean space where the nodes are exactly separated by the ECTD (see [11] for more details). Once we have computed the similarity matrix, movies are ranked according to their similarity with the user, and the closest movie that has not been watched is proposed first. Notice that, in addition, we used the same procedure as for the "PCA CT"; that is to rely on the principal components analysis subspace, but since we did not observe any improvement in comparison with $\mathbf{L}^+$ we do not show the results here.

We now introduce the other, more classical, collaborative recommendation methods to which we will compare our algorithms based on a Markov-chain model.

Nearest neighbours (kNN). The nearest-neighbour 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.

Using a nearest-neighbour technique requires a measure of "closeness" or "similarity". There is often a great deal of subjectivity involved in the choice of a similarity measure (see [6]). Important considerations include the nature of the variables (discrete, continuous, binary), scales of measurement (nominal, ordinal, interval, ratio), and specific knowledge about the subject matter.

In the case of our movie database, pairs of items are compared on the basis of the presence or absence of certain features. Similar items have more features in common than do dissimilar items. The presence or absence of a feature is described mathematically by using a binary variable, which

takes the value 1 if the feature is present (i.e., if person i has watched movie j) and the value 0 if the feature is absent (i.e., if person i has not watched movie j).

More precisely, each individual i is characterized by a binary vector $\mathbf{v}_i$ encoding the watched movies. The nearest neighbours (the most similar individuals) of individual 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 (listed in [6], p. 674) and for different k $(= 1, 2, ..., 10, 20, ..., 100)$.

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

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

where a_{pj} is the element p, j of the adjacency matrix and we keep only the k nearest neighbours for p. Notice that the best score was obtained with $k = 100$ neighbours and for the "double weights for 1-1 matches and 0-0 matches" method of [6], p. 674.

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

The predicted value of user i for item j is computed in a similar way as in the k-nearest neighbours method (see Equation 1). We again varied systematically the number of neighbours k $(= 1, 2, ..., 10, 20, ..., 100)$. This time, the best score was obtained with 60 neighbours (i.e., $k = 60$).

Katz. This similarity index has been proposed in the social sciences field and has been recently rediscovered in the context of collaborative recommendation [5] and kernel methods where it is known as the von Neumann kernel [13]. Katz proposed in [7] 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

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

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 a 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., $\alpha = 0.85 * (\text{spectral radius})^{-1}$).

Shortest path algorithm (Dijkstra). This algorithm solves a shortest path problem for a directed and connected graph with nonnegative edge weights. As a distance between two elements of the database, we compute the shortest path between these two elements.

3.4 Direct versus indirect method

Most of the ranking methods provide a proximity matrix whose elements represent proximities between each pair of nodes (i.e., a movie and a user, two users, or two movies). Therefore, there are two ways to determine the movies to suggest to a particular user (`user_i`):

(1) Use **directly** the `movies-user_i` similarities to rank all the movies for `user_i`. The movie on the top of the list is suggested first to `user_i`. It is the most natural method which will be referred to as the "*direct method*".

(2) Proceed **indirectly**, like for the nearest neighbours method: Use, in a first stage, the `users-user_i` similarities to rank all the users according to their similarity with `user_i`. The user on the top of the list is the nearest neighbour of `user_i`. In a second stage, once the k nearest neighbours are identified, the movies that are proposed to `user_i` are computed as for the k-nearest neighbours method (i.e., using Equation 1). For each method, we varied systematically the number of neighbours k $(= 1, 2, ..., 10, 20, ..., 100)$ and we keep the k giving the best degree of agreement. It will be referred to as the "*indirect method*".

Following this definition, we notice that the cosine and the k-nearest neighbours methods can only be used in the indirect way. Notice that we provide in [3] a comparison of the computing time needed by each method to provide predictions.

4 Results

4.1 Ranking procedure

Thus, for each person, we first select the movies that have not been watched. Then, we rank them according to all the described scoring algorithms. Finally, we compare the proposed rankings 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.

Notice that all the movies have not been equally watched: some movies have been watched by most users while other ones have been watched by few users. In the

Direct method										
	MaxF	CT	PCA CT	One-way	Return	$\mathbf{L}^+$	kNN	Cosine	Katz	Dijkstra
Agreement (in %)	85.69	85.66	87.08	85.64	80.65	90.99	/	/	87.90	49.11
Indirect method										
	MaxF	CT	PCA CT	One-way	Return	$\mathbf{L}^+$	kNN	Cosine	Katz	Dijkstra
Agreement (in %)	/	74.48	82.46	74.48	54.30	93.02	92.63	92.73	89.82	76.09
#Neighbours	/	100	60	100	100	100	100	60	20	100

(3)

Table 1. Degree of agreement obtained on the first experiment (preliminary results) by the various ranking procedures.

cross-validation part, the results obtained for each method differ from set to set (because of the proposed splits). Since the popularity of the movie has a significant impact on the results (the maximum frequency algorithm performs well), we decided to report two quantities in experiment 2 (cross-validation): (1) the average degree of agreement, for each ranking procedure and (2) the average difference with the degree of agreement obtained by the maximum frequency algorithm, as well as its standard deviation.

In the next section (Section 4.2), we report the results obtained for the first experiment (where the `test set` contains 10 movies for each user: preliminary results). This experiment allowed us to tune the parameters (number of neighbours, ...); we then use these optimized parameters to compute the results obtained by each method (used in the direct and indirect way) by using a 5-fold cross-validation (second experiment; Section 4.3).

4.2 Results of the first experiment (preliminary results)

Remember that, in this first experimental setting, we use a `test set` (10% of the data) in order to compute the global degree of agreement. We now detail the results obtained by all the ranking methods, for both the direct and the indirect procedures. Remember that we only report the best results obtained after having tuned the different parameters (number of neighbours, number of components, similarity measure for kNN, etc).

4.2.1 Results and discussion of the direct method

The results of the comparison are tabulated in Table 1, where we display the degree of agreement (in percent) for each method (in percent).

Based on Table 1, we observe that the best degree of agreement is obtained by the $\mathbf{L}^+$ method (90.99). The second best results (more than 3 percents behind the best method) are obtained by Katz (87.90) and PCA CT (87.08). Notice, however, that the results of the PCA CT are subject to caution, since they highly depend on the appropriate number of principal components (i.e., 60), which depends on the data set and is difficult to estimate a priori. MaxF, CT and One-way provide reasonably good results too. They present a degree of agreement of 85.69, 85.66, and 85.64 respectively. It is surprising to observe that the trivial MaxF method provides reasonably good results.

The Return method performs slightly worse than these three methods (80.65) while the worst results are obtained by Dijkstra (49.11). Indeed, it seems that nearly each movie can be reached from any person with a shortest path distance of 3. The degree of agreement is therefore close to 50% because of the large number of ties.

4.2.2 Results and discussion of the indirect method

For each method, the best result is showed in Table 1 where we display the degree of agreement (in percent) with its corresponding number of neighbours.

We observe that, for the methods that can both be used in the direct and in the indirect way, there are two trends. The degree of agreement of CT, PCA CT, and One-way decreases while for Return, $\mathbf{L}^+$, Katz and Dijkstra, on the contrary, it increases. As stated before, the MaxF method cannot be applied in the indirect way while kNN and Cosine can only be used in the indirect way.

The best degree of agreement is still obtained by $\mathbf{L}^+$ (93.02) but Cosine and kNN obtain very close results (92.73 and 92.63 respectively). Katz obtains a degree of agreement of 89.82 while the remainder of the methods range between 54.30 and 82.46.

4.3 Results of the second experiment (cross-validation)

In this second experiment, we use a 5-fold cross-validation in order to compute the global degree of agreement, for each method (used directly or indirectly). Remember that the parameters have been tuned in the first experiment; only the best results are displayed.

4.3.1 Results and discussion of the direct method

The results of the comparison are tabulated in Table 2 where we display the average degree of agreement (in percent) for each method. Table 2 also shows the average (and the standard deviation) of the differences between the results obtained by the considered method and the maximum frequency algorithm (MaxF). We observe that the best results are still obtained by $\mathbf{L}^+$ (87.23), closely followed by Katz.

Direct method										
	MaxF	CT	PCA CT	One-way	Return	$\mathbf{L}^+$	kNN	Cosine	Katz	Dijkstra
Agreement (in %)	84.07	84.09	84.04	84.08	72.63	87.23	/	/	85.83	49.96
Difference with MaxF (in %)	0	+0.02	−0.03	+0.01	−11.43	+3.16	/	/	+1.76	−34.11
STD of the difference with MaxF	0	0.01	0.76	0.01	1.06	0.84	/	/	0.24	1.52
Indirect method										
	MaxF	CT	PCA CT	One-way	Return	$\mathbf{L}^+$	kNN	Cosine	Katz	Dijkstra
Agreement (in %)	/	68.76	77.15	68.74	52.59	88.67	88.33	88.34	86.02	77.11
#Neighbours	/	100	60	100	100	100	100	60	20	100
Difference with MaxF (in %)	/	−15.31	−6.92	−15.33	−31.48	+4.60	+4.26	+4.27	+1.95	−6.96
STD of the difference with MaxF	/	1.26	0.70	1.34	2.06	1.07	0.95	1.00	0.51	1.30

(4)

Table 2. Average results (degree of agreement and difference with MaxF) obtained on the second experiment (5-fold cross-validation) by the various ranking procedures.

Three other methods give degrees of agreement close to the one of the MaxF algorithm (84.07). The Return method (72.63) and Dijkstra (49.96) obtain the worse results.

4.3.2 Results and discussion of the indirect method

For each method, the best result is showed in Table 2 where we display the degree of agreement with its corresponding number of neighbours (which is the same as in the first experiment). Table 2 also shows the average (and the standard deviation) of the differences between the results obtained by each method and the maximum frequency algorithm. We observe that the degrees of agreement of all the methods but $\mathbf{L}^+$, Katz and Dijkstra decrease when using the indirect procedure instead of the direct one. The ranking between the results of the different methods is the same as for the first experiment. However, we observe a decrease of performances (about -4 percents) in comparison with the preliminary experiment, due to the fact that 20% of the data (instead of 10%) are now used for performance evaluation, and the number of movies included in the test set for each person is arbitrary.

5 Conclusions and further work

We introduced a general procedure for computing similarities between elements of a database. These similarity measures can be used in order to compare items belonging to database tables that are not necessarily directly connected. It relies on the degree of connectivity between these items. We showed through experiments performed on the MovieLens database that some of these quantities perform well in comparison with standard methods. The main drawback of this method is that it does not scale well for large databases: the Markov model has as many states as elements in the database. For large databases, we have to rely on iterative formula and on the sparseness of the matrix. We are now comparing the different kernels on graph that have been proposed in the literature on the same collaborative recommendation task. We are also working on other generalizations of standard multivariate statistics methods to the mining of databases, such as discriminant and canonical correlation analysis.

References

[1] F. Buckley and F. Harary. *Distance in graphs*. Addison-Wesley Publishing Company, 1990.

[2] S. Dzeroski. Multi-relational data mining: an introduction. *ACM SIGKDD Explorations Newsletter*, 5(1):1–16, 2003.

[3] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Markov-chain computation of similarities between nodes of a graph, with application to collaborative filtering. *Submitted for publication, http://www.isys.ucl.ac.be/staff/francois/Articles/Saerens2005a.pdf*, 2005.

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

[5] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems*, 22(1):116–142, 2004.

[6] R. Johnson and D. Wichern. *Applied Multivariate Statistical Analysis, 5th Ed.* Prentice Hall, 2002.

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

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

[9] A. Langville and C. Meyer. A survey of eigenvector methods of web information retrieval. *The SIAM Review*, 47(1):135–161, 2005.

[10] J. Norris. *Markov Chains*. Cambridge University Press, 1997.

[11] M. Saerens, F. Fouss, L. Yen, and P. Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. *Proceedings of the 15th European Conference on Machine Learning (ECML 2004)*, pages 371–383, 2004.

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

[13] B. Scholkopf and A. Smola. *Learning with kernels*. The MIT Press, 2002.

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

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