(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 July 2007 (05.07.2007)

PCT

(10) International Publication Number
WO 2007/075622 A2

(51) International Patent Classification:
G06F 7/00 (2006.01)

(21) International Application Number:
PCT/US2006/048330

(22) International Filing Date:
19 December 2006 (19.12.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/752,102 19 December 2005 (19.12.2005) US

(71) Applicant *(for all designated States except US)*: **MUSICSTRANDS, INC.** [US/US]; 760 S.w. Madison, Suite 106, Corvallis, Oregon 97333 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **TORRENS, Marc** [ES/US]; 760 S.W. Madison Street, Suite 106, Corvallis, Oregon 97333 (US). **FERRERA, Pere** [ES/ES]; 8193 Bellaterra, E-08193 Barcelona (ES).

(74) Agent: **THAYNE, Matthew, D.**; One Utah Center, 201 So. Main Street, Suite 1100, Salt Lake City, Utah 84111 (US).

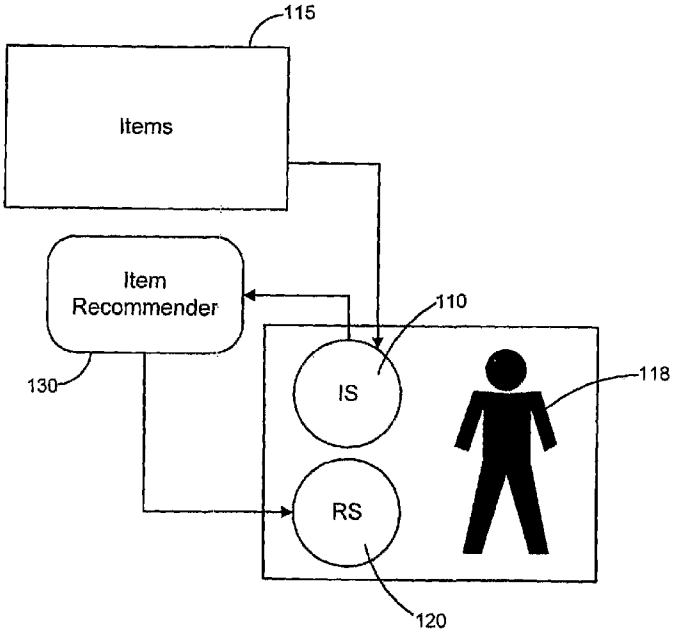
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: USER-TO-USER RECOMMENDER



(57) **Abstract:** Disclosed are embodiments of systems and methods for recommending relevant users to other users in a user community. In one implementation of such a method, two different sets of data are considered: a) music (or other items) that users have been listening to (or otherwise engaging), and b) music (or other items) recommendations that users have been given. In some embodiments, pre-computation methods allow the system to efficiently compare item sets and recommended item sets among the users in the community. Such comparisons may also comprise metrics that the system can use to figure out which users should be recommended for a given target user.

WO 2007/075622 A2

USER-TO-USER RECOMMENDER

Brief Description of the Drawings

[0001] Understanding that drawings depict only certain preferred embodiments of the invention and are therefore not to be considered limiting of its scope, the preferred embodiments will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

[0002] FIG. 1 is a diagram showing the basic components and sources for a user profile according to one embodiment.

[0003] FIG. 2 depicts a graph showing the position of a target user "X" in ADG Space according to one embodiment.

[0004] FIG. 3 is a diagram showing a basic architecture schema for a user recommender according to one embodiment.

[0005] FIG. 4 depicts a qualitative scale indicative of the relevance of particular items to a particular user within a user community.

[0006] FIG. 5 is a diagram showing a Servlet View of one embodiment of a user recommender system.

[0007] FIG. 6 is a diagram showing a Recommender View of one embodiment of a user recommender system.

[0008] FIG. 7 is a diagram showing a Manager View of one embodiment of a user recommender system.

[0009] FIG. 8 is a core Unified Modeling Language (UML) diagram of one embodiment of a user recommender system.

[0010] FIG. 9 is a diagram depicting an example of the information that can be extracted from a GraphPlotter tool used with one embodiment of a user recommender system.

[0011] FIG. 10 is a graph representing relationships between types of listeners according to a "Frequency/Knowledge" model.

Detailed Description of Preferred Embodiments

[0012] In the following description, certain specific details of programming, software modules, user selections, network transactions, database queries, database structures, etc., are provided for a thorough understanding of the specific preferred embodiments of the invention. However, those skilled in the art will

recognize that embodiments can be practiced without one or more of the specific details, or with other methods, components, materials, etc.

[0013] In some cases, well-known structures, materials, or operations are not shown or described in detail in order to avoid obscuring aspects of the preferred embodiments. Furthermore, the described features, structures, or characteristics may be combined in any suitable manner in a variety of alternative embodiments. In some embodiments, the methodologies and systems described herein may be carried out using one or more digital processors, such as the types of microprocessors that are commonly found in PC's, laptops, PDA's and all manner of other desktop or portable electronic appliances.

[0014] Disclosed are embodiments of systems and methods for recommending users to other users in a user community. As used herein, a "user recommender" is a module integrated in a community of users, the main function of which is to recommend users to other users in that community. There may be a set of items in the community for the users of the community to interact with. There may also be an item recommender to recommend other items to the users. Examples of recommender systems that may be used in connection with the embodiments set forth herein are described in U.S. Patent Publication No. 2006-0184558 titled "Recommender System for Identifying a New Set of Media Items Responsive to an Input Set of Media Items and Knowledge Base Metrics," and U.S. Patent Publication No. 2006-0173910 titled "Dynamic Identification of a New Set of Media Items Responsive to an Input Mediaset."

[0015] As used herein, the term "media data item" is intended to encompass any media item or representation of a media item. A "media item" is intended to encompass any type of media file which can be represented in a digital media format, such as a song, movie, picture, e-book, newspaper, segment of a TV/radio program, game, etc. Thus, it is intended that the term "media data item" encompass, for example, playable media item files (*e.g.*, an MP3 file), as well as metadata that identifies a playable media file (*e.g.*, metadata that identifies an MP3 file). It should therefore be apparent that in any embodiment providing a process, step, or system using "media items," that process, step, or system may instead use a representation of a media item (such as metadata), and vice versa.

[0016] The user recommender may be capable of selecting relevant users for a given target user. To do so, users should be comparable entities. The component

that defines a user in a community may be referred to as the user profile. Thus, a user profile may be defined by defining two sets, such that comparing two users will be a matter of intersecting their user profile sets. For example, with reference to FIG. 1, the first set may be the "items set," referenced at 110 in FIG. 1, which may contain the most relevant items 115 for a particular user 118. The second set may be the "recommendations set," referenced at 120 in FIG. 1, which may contain the most relevant recommended items for user 118. The items set 110 can be deduced by the item usage and/or interaction of a certain user with certain items, whereas the recommendations set can be deduced by using an item recommender 130. In some cases, the items set 110 can be used as the input for the recommender 130, thereby obtaining a recommendations set as the output.

[0017] The items described in the following examples and implementations will be for musical or other media items. However, it should be understood that the implementations described herein are not item-specific and may operate with any other type of item used/shared by a community of users.

[0018] For musical or multimedia items (tracks, artists, albums, etc.), users may interact with the items by using them (listening, purchasing, etc.). The sets in such embodiments will be referred to as "musical sets," as they contain musical items. These sets will therefore be referred to as the "Music Set" and the "Recommendations Set."

[0019] The Music Set is the musical set formed by the items the user is listening to. A User A's Music Set will be denoted herein as M_a.

[0020] The Recommendations Set is the musical set formed by the items the user is being recommended. A User A's Recommendations Set will be denoted herein as R_a.

[0021] To compare two user profiles, the intersection between one or more of their respective sets may be analyzed. A variety of different metrics may also be applied to set intersections to provide useful data. Some such metrics will describe relations between users. For example, four elementary intersecting cases are:

[0022] $M_a \cap M_b$, $M_a \cap R_b$, $R_a \cap M_b$, and $R_a \cap R_b$. Analyzing these cases may lead to complex cases that may be labeled or classified as different relations. For example, in one implementation, four relevant relations may be extracted:

[0023] Peer : If M_a intersects M_b sufficiently, B is considered a "Peer" of A.

[0024] Guru: If M_a intersects R_b sufficiently, B is considered a "Guru" of A.

[0025] Peer-Guru: Peer condition plus Guru condition. B is considered a "Peer-Guru" of A.

[0026] Follower : If R_a intersects M_b sufficiently, B is considered a "Follower" of A.

[0027] Peer relation may be relevant because it gives the target user another user whose musical (or other item) library is similar in some way to the target user's musical library. Guru relation may be relevant because it gives the target user another user whose musical library contains music the target user may enjoy discovering. Peer-Guru may be relevant because it gives the target user both the affinity and the discovery experiences of the Peer and Guru relations, respectively, toward one or more recommended users. Follower relation may be relevant because it gives the user the chance to know which users may be influenced by him or her.

[0028] Illustrative concrete metrics will now be disclosed, from which the aforementioned relations, for example, can be deduced. A metric may be a function that takes as input two (or more) user profiles and produces a measurable result as an output. The metrics discussed below are unidirectional, meaning that the order of the parameters can change the result.

[0029] The "Affinity" metric answers the question, "How much does M_a intersect with M_b?" In other words, how much "affinity experience" does user A have towards user B?

[0030] The "Discovery" metric answers the question, "How much does R_a intersect with M_b?" In other words, how much "discovery experience" does user A have towards user B?

[0031] The "Guidance" metric answers the question, "How much does M_a intersect with R_b?" In other words, how much can user A guide user B?

[0032] With these metrics, Peer relations can be found by maximizing the Affinity metric, Guru relations can be found by maximizing the Discovery metric, Peer-Guru relations can be found by maximizing both the Affinity and the Discovery metric, and Follower relations can be found by maximizing the Guidance metric. A total relevance of one user toward another user can be computed, for example, by defining a function that operates with each (or greater than one) of the metrics. For a target user, all the other users in the community can be located as points into a three-dimensional space ("ADG Space") where X = Affinity, Y = Discovery, and Z = Guidance. Defining the metrics as to return a number between [0,1], all the users in

the community can be enclosed within a cube of 1x1x1 in that space. FIG. 2 illustrates the position of a sample target user "X" in ADG Space.

[0033] To implement a user recommender following the conceptual model explained, an under-lying system may be built. One such system may be configured such that:

[0034] 1. There is a user community and an item recommender from which from which a user profile for each user in the community can be extracted. This information may be fed by one or more data sources.

[0035] 2. There is an implementation of the user recommender that builds the data and the operations of the model, and collects the data from the data sources.

[0036] A basic architecture schema for a user recommender according to one implementation is shown in FIG. 3. As shown in the figure, data sources 310 provide data to (and receive data from) a user community 320. Data sources 310 may also provide a data feed to a user recommender 330. User recommender 330 interacts with the user community 320. In particular, requests for user recommendations may be received from the user community 320 and recommended users may, in turn, be provided to the user community 320. Of course, it is contemplated that some implementations may rely upon receiving requests from users to generate recommended user sets and other implementations may generate such sets and offer them to users in the community without first receiving user requests.

[0037] It may also be desirable to provide a scalable architecture solution. Given a request, it may not be feasible to compare the target user to all of the users in the community (the response time may grow linearly with the number of users). A number of solutions to this problem may be implemented. For example:

[0038] 1. The user data may be clusterized and the target user compared to the right cluster.

[0039] 2. A fixed number or subset of users may be selected from the user community. This subset of users may be referred to as "Recommendable Users" and the target user(s) may be compared to that size-fixed set. The Recommendable Users may be selected by some procedure that allows the system to recommend the most interesting users in the community.

[0040] A musical set entity can be modeled as a sparse vector of an N-dimensional space where N is the total number of musical items in our universe. Each dimension refers to a different item, whereas each concrete value refers to the

relevance of that item. Adding a relevance value for each item allows the underlying system or implementation to be aware of the most relevant items for a certain user.

[0041] In some implementations, the items can be music tracks. However, in such embodiments, intersections between items in two user's sets may be less probable (due to a sparsity problem). In addition, such intersections may be computationally expensive.

[0042] These issues may be addressed in some embodiments by instead working on the artist level. The probability of intersection of artists instead of tracks is higher. On the other hand, the relevance value may depend on the data source from which the items are extracted. A normalization process may therefore be used so that all the relevance values finally belong to a known value scale, such as a qualitative value scale.

[0043] For example, FIG. 4 depicts a qualitative scale indicative of the relevance of particular items, such as artists, to a particular user. Artists (or other items) with a relevance value of less than C1 for a given user will be considered "Low," those with a relevance value between C1 and C2 will be considered "Medium," those with a relevance value between C2 and C3 will be considered "Medium/High," and those with a relevance value greater than C3 will be considered "High."

[0044] Details and examples of an illustrative normalization process are discussed later, along with other approaches for finding the relevance of a certain item for a certain user.

[0045] A user entity can be modeled as an entity with an unique ID plus two musical set entities, so that we have all the data needed to compute intersections according to the conceptual model discussed herein.

[0046] Some operations that may be implemented in some embodiments of the invention will now be discussed. The primitive operations are those that are needed to compare two user entities, and that involve intersections between musical sets. For example:

[0047] 1. The size of a musical set can be represented as:

$$|M_u| = \sum_{k=1}^{N} M_{uk}$$

Where M_{uk} is the relevance value of the item k in the set M_u.

[0048] 2. The size of an intersection can be represented as:

$$|M_u \cap M_{u'}| = \sum_{k=1}^{M} min(M_{uk}, M_{u'k})$$

For all those *M* items that are in common in M_u and $M_{u'}$.

[0049] 3. The Affinity, Discovery, and Guidance metrics can be represented as follows.

[0050] One approach to the Affinity metric consists of calculating the size of M_u with $M_{u'}$ and normalizing it by the size of M_u, as follows:

$$Affinity(U, U') = \frac{|M_u \cap M'_u|}{|M_u|}$$

[0051] As another possibility, if we consider that the intersection of R_u and $R_{u'}$ is somehow an affinity measure, then we can add this factor to the whole formula, weighting it by a *K* factor and thereby normalizing the measure:

$$Affinity(U, U') = \frac{\frac{|M_u \cap M_{u'}|}{|M_u|} + \frac{|R_u \cap R_{u'}|}{K \cdot |R_u|}}{1 + \frac{1}{K}}$$

[0052] Note that a high Affinity of U to U' does not necessarily mean a high Affinity of U' to U.

[0053] Corresponding formulas for Discovery and Guidance are as follows:

$$Discovery(U, U') = \frac{|R_u \cap M'_u|}{|R_u|}$$

$$Guidance(U, U') = \frac{|M_u \cap R'_u|}{|R'_u|}$$

[0054] Note that it is always true that *Discovery*(*U*, *U'*) = *Guidance*(*U'*, *U*).

[0055] The following model operations are illustrative global operations that may be implemented in the user recommender that, by means of using primitive operations, allow it to compute the desired result.

[0056] – getBestUsers(User, Requirement):

[0057] By computing a certain set of metrics, a set of recommended users may be returned for the target user. The Requirement may specify what kind of users are to be recommended and what metrics are to be considered. A general algorithm for this function may be as follows:

[0058] 1. Let TU be the Target User, RUS the Recommendable Users Set and REQ the Requirement of the request.

[0059] 2. For each User U in RUS, compute the necessary Metrics (TU,U) according to REQ and store the result, together with the compared user U of RUS.

[0060] 3. Sort RUS by the result of the comparision so that at the beginning of the list we have the best users according to REQ.

[0061] 4. Return a sublist of RUS, starting at the beginning.

[0062] – getRelevance(User1, User2):

[0063] By computing all the metrics of User1 toward User2, a floating number may be returned by computing a function performing some calculation with all the metric values, which answers the question: How relevant is User2 for User1? This function may, for example, calculate the length of the vector in the ADG Space.

[0064] The user recommender may be implemented as a Java Web Module. This module may be deployed, for example, as a webapp in a Tomcat environment. In one implementation, the Data Sources for such an implementation may be as follows:

[0065] 1. "Reach" API: Returns playcount data for each user. Some implementations may be able to deduce a musical set from this data.

[0066] 2. "UMA" Recommender: Returns recommended items for a set of items. Some implementations may be able to deduce a musical set from this data using as input, for example, the Music Set of the user profile, and consequently obtaining the Recommendations Set of the user profile explained above.

[0067] 3. "KillBill" API: Returns some extra information about a user, for example, its alias.

[0068] One scalable solution is to obtain the best N users of the community through the Reach API and make them Recommendable Users. In some implementations, a CLUTO clustering program may be used. CLUTO programs are open source software, and are available for download at: <http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download>. In other implementations, a WEKA clustering program may be used. WEKA is also an open source program, and is available for download at: <http://www.cs.waikato.ac.nz/ml/weka/>. A file-cache system may also be built and used in some implementations.

[0069] Therefore, only the best users may be recommended, even though recommendations may be provided to all the users. If a user is not in the best N set,

then the user recommender may ask in real-time for the user profile of that user to be added to the data sources.

[0070] A Java implementation may consist of a set of classes the logic of which can be partitioned as follows:

[0071] Servlet View: Where all the servlet request/response logic is enclosed.

[0072] Recommender View: Where the main operations are performed by the singleton Recommender, often delegating them to a Manager.

[0073] Manager View: Where all the operations are actually performed, by using the Core classes.

[0074] Core View: Where the foundations of the model are established by a set of Java Classes.

[0075] The aforementioned "Views" will now be described in greater detail. The user recommender may be implemented as a set of HTTP Servlets. The recommender may be implemented as a Singleton Instance, which may be used by three different servlets, as shown in FIG. 5.

[0076] 1. The Debug Servlet 500 attends debug commands. For instance, action=stats may perform a plot of the recommender 530 internal memory statistics.

[0077] 2. The Recommender Servlet 510 attends the recommendation requests according to the model operations explained previously.

[0078] 3. The Update Servlet 520 performs the update process on the user data.

[0079] The Recommender may be a singleton class mainly formed by a manager and a request cache. Concurrent accesses to the cache may also be controlled by semaphores to avoid inconsistent information. For example, if the Debug Servlet sends a flush cache command, if there is a recommendation request waiting to get the result from cache, a null response can be received if the request is processed after the flush has been performed.

[0080] In some implementations, the general algorithm for accessing the request cache is as follows:

[0081] 1. Close the semaphore if it is opened; otherwise wait.

[0082] 2. Is the result in the cache?

[0083] 3. If so, take the result out from the cache.

[0084] 4. Open the cache semaphore.

[0085] The Cache may be implemented in a Hash Map, the keys of which are string request hashes and the values of which are a request result. The string

request hashes may be calculated to obtain a unique string for each set of parameters that may produce a different result.

[0086] FIG. 6 shows an example of a user to user recommender 600 and its basic interactions with other elements of the system. Recommender 600 has a Manager 610 and a Request Cache 620. Manager 610 has one or more external connections, shown at 612, and a data feed, shown at 614.

[0087] The Manager may be implemented as a singleton class with two layers—the service layer and the dataspace layer, as shown at 700 in FIG. 7.

[0088] The service layer 710 may contain singleton instances of services 712. Each service 712 may be used to perform a set of tasks of a particular type. For example, an "Update Service," containing all the logic for performing the update on the data, may be provided. A "Comparator Service," containing all the logic for performing comparators between the Data, may also be provided. These services may communicate directly with the dataspace layer 720.

[0089] The dataspace layer 720 may contain all the connections to external services (such as UMA 722, KillBill 724, and Reach 726) as well as the main memory structures 728 where the recommendable users are permanently stored.

[0090] The basic classes for some implementations are: User 810, MeasuredUser 820, UserSimilarity 830, and Requirement 840, as shown in FIG. 8. In such implementations, a Musical Set 850 for a user 810 may be implemented as a set of pairs (Item 854 and Relevance 856). Certain implementations of Item and Relevance are ArtistItem (containing Artist ID's) and SimpleRelevance (a magnitude between 1 and 3). There may also be an ItemFactory and a RelevanceFactory, the tasks of which are to create Item and Relevance objects, taking the name of the implementation as input. In such embodiments, the implementation of Item and Relevance can easily be changed at any stage of the project without affecting the other core classes.

[0091] When a user is compared to another user, we have a MeasuredUser 820, which is the compared user together with UserSimilarity 830 instances. A UserSimilarity 830 specifies how much of each Metric 860 is correlated with the target User. The client may also be able to specify a Requirement to maximize/minimize different metrics. The Affinity, Discovery, and Guidance Metrics are shown in FIG. 8 at 862, 864, and 866, respectively.

[0092] As described previously, the Affinity, Discovery and Guidance metrics may be implemented, but other metrics can also be implemented by extending the interface metric.

[0093] The interface metric specifies that each metric has to perform an intersection between two users, the result of which is parametrizable. The interface may also specify that each metric has to return a computation value measuring the relevance between two users according to the metric. This value may also be parameterizable, so that each metric is double-parameterized. The computation value may also be normalized between 0 and 1. As an example, a new metric "Age Affinity" could be implemented. This Metric might return a signed Integer comprising the difference between two user ages as the intersection and/or a String representing the qualitative age difference of one user to the other ("younger", "much younger," etc.). The normalized computation might be calculated so that 1 means the two users have the same age and 0 means the two users are too far apart in age to be considered related for purposes of the system.

[0094] In one example, a webapp was deployed in two production machines controlled by a load balancer. The number of recommendable users was about 1000 and the system was able to respond to more than 500 requests per day. A stress test was made with Apache Jmeter, a Java desktop application available for download at: <http://jakarta.apache.org/site/downloads/downloads_jmeter.cgi>. In this test, several requests were sent to the server, and the response time increased linearly with the number of simultaneous requests. The test result numbers were as follows:

Requests	Requests per second	Average Processing Time	Average Response Time
100	1	60 ms	290 ms
200	2	110 ms	370 ms
1000	10	120 ms	420 ms
10000	100	500 ms	1800 ms

[0095] In some implementations, a viewable graph can be plotted by using a GraphPlotter tool. FIG. 9 provides an example of the information that can be extracted from a GraphPlotter tool. As shown in FIG. 9, Discovery and/or Affinity values between various users in the user community are represented in the graph.

[0096] Additional details of particular implementations will now be described in greater detail. Let's suppose we have two users, A and B, with Music and Recommendation sets M_a, R_a and M_b, R_b, respectively. If R is the output of an item recommender generated from input M, for some item recommenders it is always true that

$$M \cap R = \varnothing$$

[0097] As set forth previously, the four possible intersections are $M_a \cap M_b$, $M_a \cap R_b$, $R_a \cap M_b$, and $R_a \cap R_b$. The total number of cases is 12:

[0098] Peer relation: $M_a \cap M_b$. A and B have common musical tastes.

[0099] Peer-Brother relation: $M_a \cap M_b + R_a \cap R_b$. A and B have common musical tastes and may also have common musical tastes in the future.

[00100] Guru-follower relation: $M_a \cap R_b$. B can learn from A (A is a guru to B and B is a follower of A).

[00101] Hidden-peer relation: $R_a \cap R_b$. A and B may evolve to common musical tastes.

[00102] Peer-guru/Peer-follower relation: $M_a \cap M_b + M_a \cap R_b$. B can learn from A, but B has already learned something from A. This case may be treated as a special case of Peer or as a special case of Guru-follower. If treated as the first, then we can say that this is a "stronger" Peer (the second condition assures that the next "state" of user B's taste is also a Peer state between A and B), whereas if treated as the second, then it may be considered a "weaker" Guru-Follower relation (the follower will see some of his music in the Guru's music).

[00103] Peer-Brother-Guru/Peer-Brother-Follower relation: $M_a \cap M_b + M_a \cap R_b + R_a \cap R_b$. The same as above, but with intersection in recommendations.

[00104] Static Guru-Follower relation: $M_a \cap R_b + R_a \cap R_b$. B can learn from A and B will still learn from A if A moves towards the next state. It is a stronger case of Guru-Follower.

[00105] Crossing-trains relation: $M_a \cap R_b + R_a \cap M_b$. B learns from A and A learns from B. However, these users' next states are not going to intersect, so this is a strange case of Guru-Follower (because of being bidirectional).

[00106] Taxi Relation: $M_a \cap R_b + R_a \cap M_b + M_a \cap M_b$. The same as above but with intersection in music.

[00107] Meeting-trains relation: $M_a \cap R_b + R_a \cap M_b + R_a \cap R_b$. B learns from A, A learns from B, and their next state is going to intersect. If A or B moves to the next state, the other can still learn from him. If both move, then they are going to be Peers. This may be the strongest case of bidirectional Guru-Follower.

[00108] Perfect Connection Relation: $M_a \cap R_b + R_a \cap M_b + M_a \cap M_b + R_a \cap R_b$. Everything intersects.

[00109] There may also be ways for determining/calculating how relevant an artist is for a particular user. For example, if the system has playcounts of the artist for the user, the data may be normalized by setting absolute cut-off points such that certain numbers of playcounts can be considered "Low," certain other numbers of playcounts can be considered "Medium," and so on.

[00110] Alternatively, if the system has a set of playlists for the user, the number of times the artist appears in the playlists may be counted. The methodology may then proceed as described above (*i.e.*, with cut-off points).

[00111] As another alternative, if the system has a recommended set, the relevance of the artist based on the position it occupies in the recommended list may be calculated and used in the analysis. Of course, this assumes that the recommender provides a ranked list of recommended artists.

[00112] In some implementations, users may further be classified by how frequently they listen to a given artist, how many songs the user has in his or her profile from the artist, and/or otherwise how familiar the user is with a given artist. For example, for each artist a user listens to, we have:

[00113] 1. *F*: Frequency of listening; and

[00114] 2. *K*: Knowledge (how many songs from this artist the user knows).

[00115] The values for *F* and *K* can be classified as *High* or *Low*. Listeners for a particular artist can therefore be classified as:

Listener	Frequency	Knowledge
A	Low	Low
B	Low	High
C	High	Low
D	High	High

[00116] In general, only listeners of the same type will match, but if we imagine these classifications as points on a perfect square (where 0 is Low and 1 is High), A

is distance 1 to B and C, and distance $\sqrt{2}$ to D. Likewise, B is distance 1 to A and D, and distance $\sqrt{2}$ to C, and so on.

[00117] However, it may the case that the frequency of listening is not as relevant as the knowledge. So one dimension can be made larger than the other, which makes the square become larger around the K dimension.

[00118] With this approach, A is *High* close to C, *Medium* close to B and *Low* close to D. These relationships are represented graphically in FIG. 10. The Relevance of an artist A for a given user U may therefore be provided as:

$$Rel(U,A) = (1 + K(U,A))^2 + f(U,A)$$

where K(A) $\subseteq$ [0, 1] is a function that measures the Knowledge a User U has about Artist A, and f(A) $\subseteq$ [0, 1] is a function returning the relative frequency of User U listening to this Artist.

[00119] In some embodiments, K can be deduced from n/N where n is the number of songs from a certain artist that a user knows, and N is the total songs for this artist. F can likewise be deduced from n/P where n is the number of playcounts from a certain artist that a user has listened to, and P is the total of playcounts for this user. F may be computed through the Reach API (described above) in some implementations.

[00120] The above description fully discloses the invention including preferred embodiments thereof. Without further elaboration, it is believed that one skilled in the art can use the preceding description to utilize the invention to its fullest extent. Therefore the examples and embodiments disclosed herein are to be construed as merely illustrative and not a limitation of the scope of the present invention in any way.

[00121] It will be obvious to those having skill in the art that many changes may be made to the details of the above-described embodiments without departing from the underlying principles of the invention. Therefore, it is to be understood that the invention is not to be limited to the specific embodiments disclosed and that modifications and other embodiments are intended to be included within the scope of the appended claims.

[00122] The scope of the present invention should, therefore, be determined only by the following claims.

Claims

1. A method for recommending users in a user community, the method comprising:

selecting a first user within the user community;

selecting a user set within the user community;

comparing a user profile for the first user with user profiles for each of the users in the user set; and

generating a recommended user set for the first user; wherein the recommended user set comprises at least one user within the user community.

2. The method of claim 1, wherein the recommended user set comprises a plurality of users within the user community.

3. The method of claim 1, wherein the step of selecting a first user comprises receiving a request from the first user to generate the recommended user set.

4. The method of claim 1, wherein the user set comprises each of the users in the user community.

5. The method of claim 1, wherein the user profile comprises a media item set including a plurality of media data items.

6. The method of claim 5, wherein the media data items comprise metadata identifying a plurality of media items.

7. The method of claim 5, wherein the step of generating a recommended user set comprises analyzing playcounts for media items in the media item set.

8. The method of claim 5, wherein the user profile further comprises a recommended media item set including a plurality of recommended media data items.

9. The method of claim 8, wherein the recommended user set is generated by using at least one of a metric that analyzes the intersection of the media item set with a media item set of media data items for each of the users in the user set, a metric that analyzes the intersection of the media item set with a recommended media item set of recommended media data items for each of the users in the user set, a metric that analyzes the intersection of the recommended media item set with a media item set of media data items for each of the users in the user set, and a metric that analyzes the intersection of the recommended media item set with a

recommended media item set of recommended media data items for each of the users in the user set.

10. The method of claim 9, wherein the recommended user set is generated by further using playcounts for media items in the media item set for each of the users in the user set.

11. The method of claim 8, wherein the recommended user set is generated by using metric that analyzes the intersection of the media item set with a media item set of media data items for each of the users in the user set, a metric that analyzes the intersection of the media item set with a recommended media item set of recommended media data items for each of the users in the user set, a metric that analyzes the intersection of the recommended media item set with a media item set of media data items for each of the users in the user set, and a metric that analyzes the intersection of the recommended media item set with a recommended media item set of recommended media data items for each of the users in the user set.

12. A computer-readable medium having stored thereon computer executable instructions for performing a method for recommending users in a user community, the method comprising:

selecting a first user within the user community;

selecting a user set within the user community;

comparing a user profile for the first user with user profiles for each of the users in the user set; and

generating a recommended user set for the first user, wherein the recommended user set comprises at least one user within the user community.

13. The computer-readable medium of claim 12, wherein the step of selecting a first user comprises receiving a request from the first user to generate the recommended user set.

14. The computer-readable medium of claim 12, wherein the user profile comprises a media item set including a plurality of media data items.

15. The computer-readable medium of claim 14, wherein the media data items comprise metadata identifying a plurality of media items.

16. The computer-readable medium of claim 14, wherein the step of generating a recommended user set comprises analyzing playcounts for media items in the media item set.

17. The computer-readable medium of claim 14, wherein the user profile further comprises a recommended media item set of recommended media data items.

18. The computer-readable medium of claim 17, wherein the recommended user set is generated by using at least one of a metric that analyzes the intersection of the media item set with a media item set of media data items for each of the users in the user set, a metric that analyzes the intersection of the media item set with a recommended media item set of recommended media data items for each of the users in the user set, a metric that analyzes the intersection of the recommended media item set with a media item set of media data items for each of the users in the user set, and a metric that analyzes the intersection of the recommended media item set with a recommended media item set of recommended media data items for each of the users in the user set.

19. The computer-readable medium of claim 18, wherein the recommended user set is generated by further using playcounts for media items in the media item set for each of the users in the user set.

20. The computer-readable medium of claim 17, wherein the recommended user set is generated by using metric that analyzes the intersection of the media item set with a media item set of media data items for each of the users in the user set, a metric that analyzes the intersection of the media item set with a recommended media item set of recommended media data items for each of the users in the user set, a metric that analyzes the intersection of the recommended media item set with a media item set of media data items for each of the users in the user set, and a metric that analyzes the intersection of the recommended media item set with a recommended media item set of recommended media data items for each of the users in the user set.

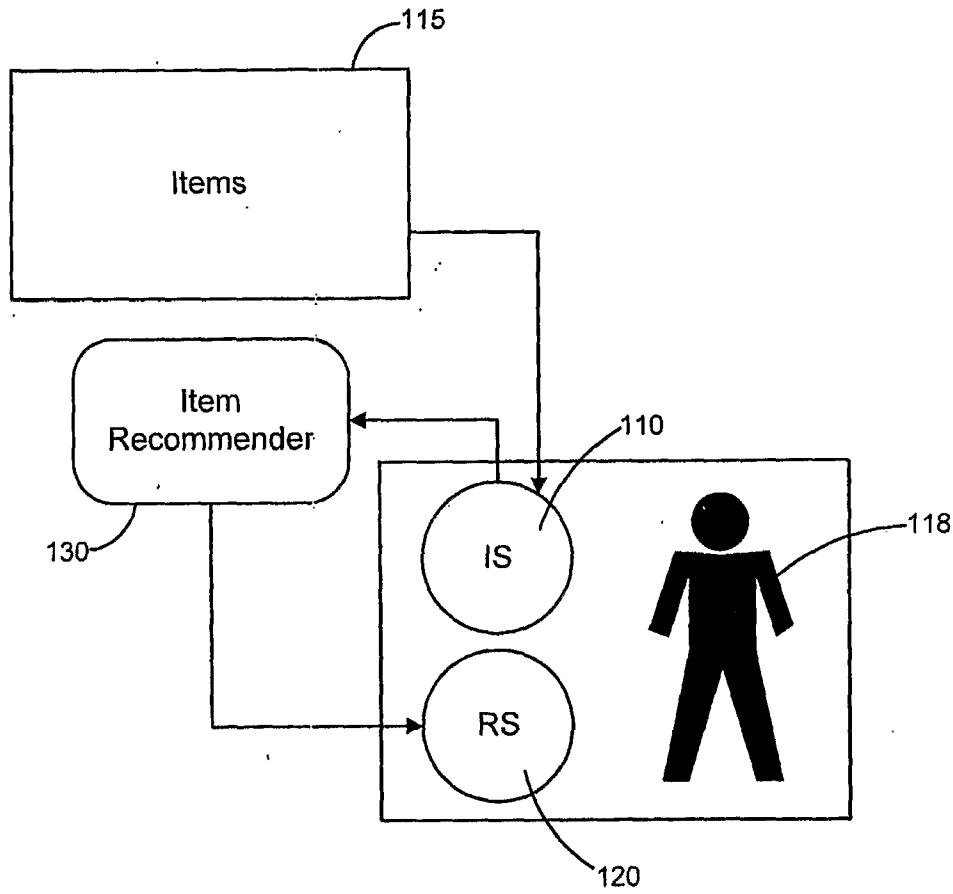


FIG. 1

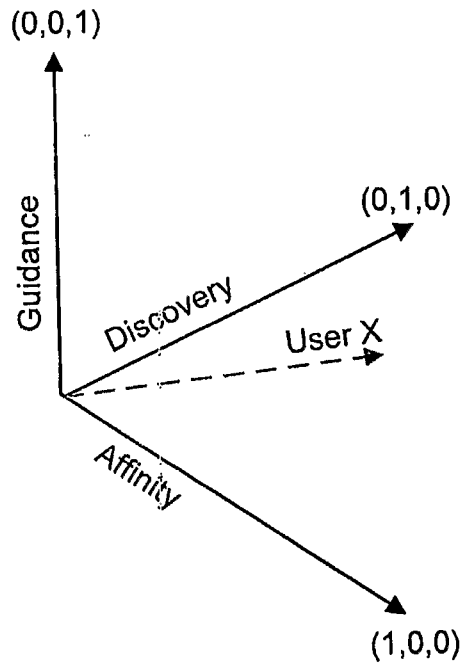


FIG. 2

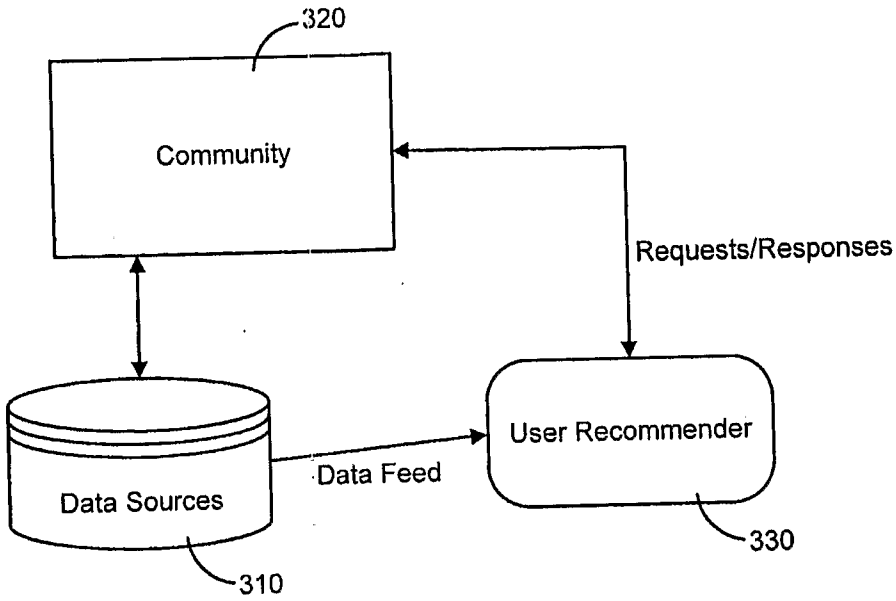

320
Community
Requests/Responses
Data Sources
Data Feed
User Recommender
310
330

FIG. 3

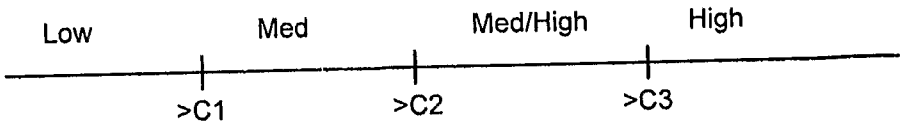


FIG. 4

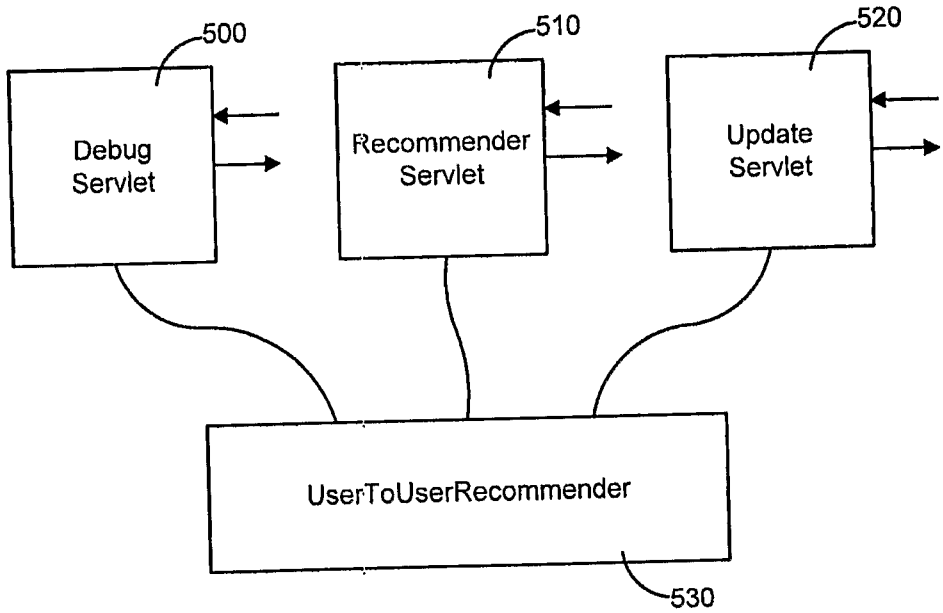

500
510
520
Debug Servlet
Recommender Servlet
Update Servlet
UserToUserRecommender
530

FIG. 5

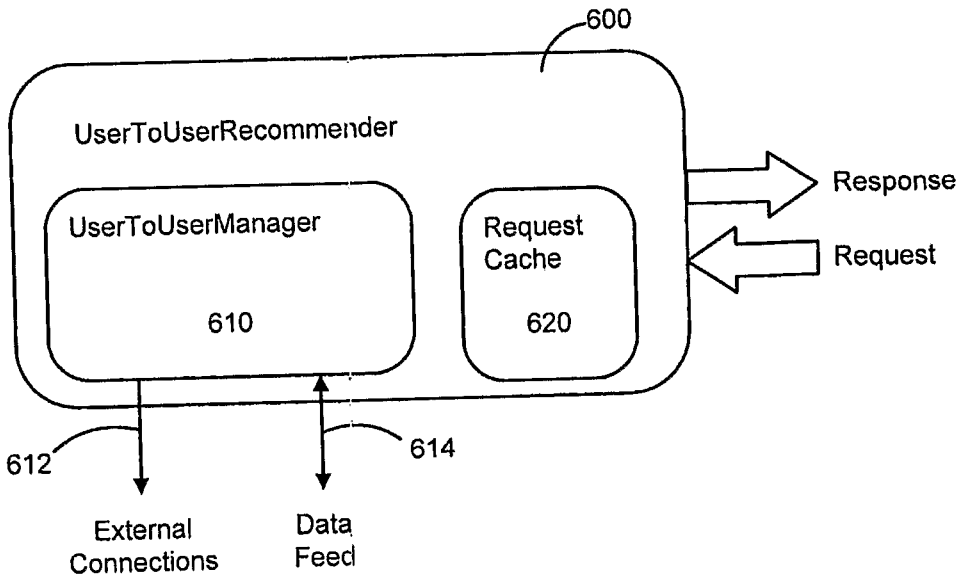

600
UserToUserRecommender
UserToUserManager
610
Request
Cache
620
Response
Request
612
614
External
Connections
Data
Feed

FIG. 6

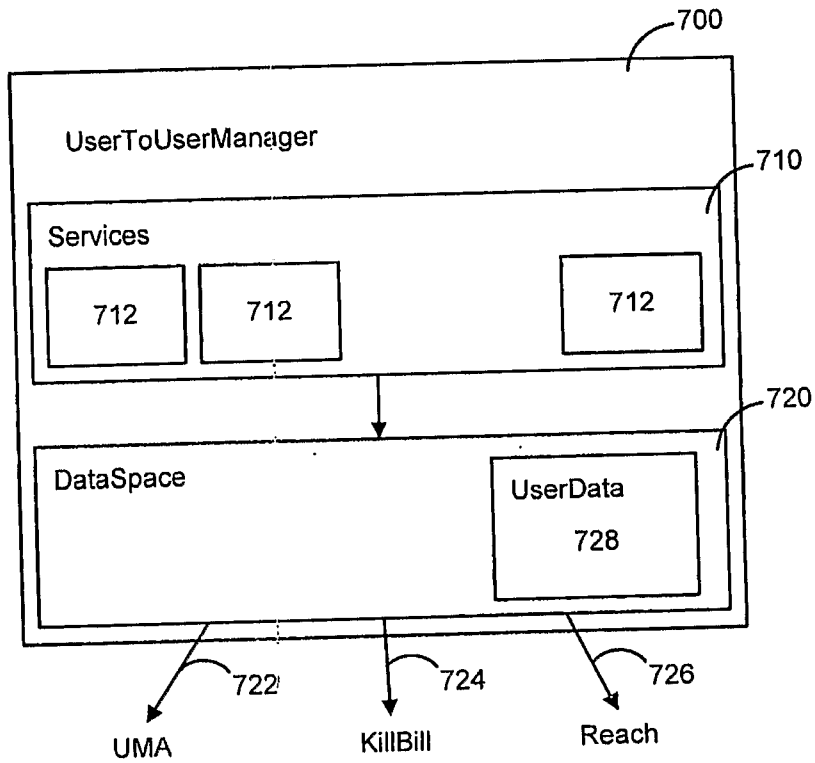

700
UserToUserManager
710
Services
712
712
712
720
DataSpace
UserData
728
722
724
726
UMA
KillBill
Reach

FIG. 7

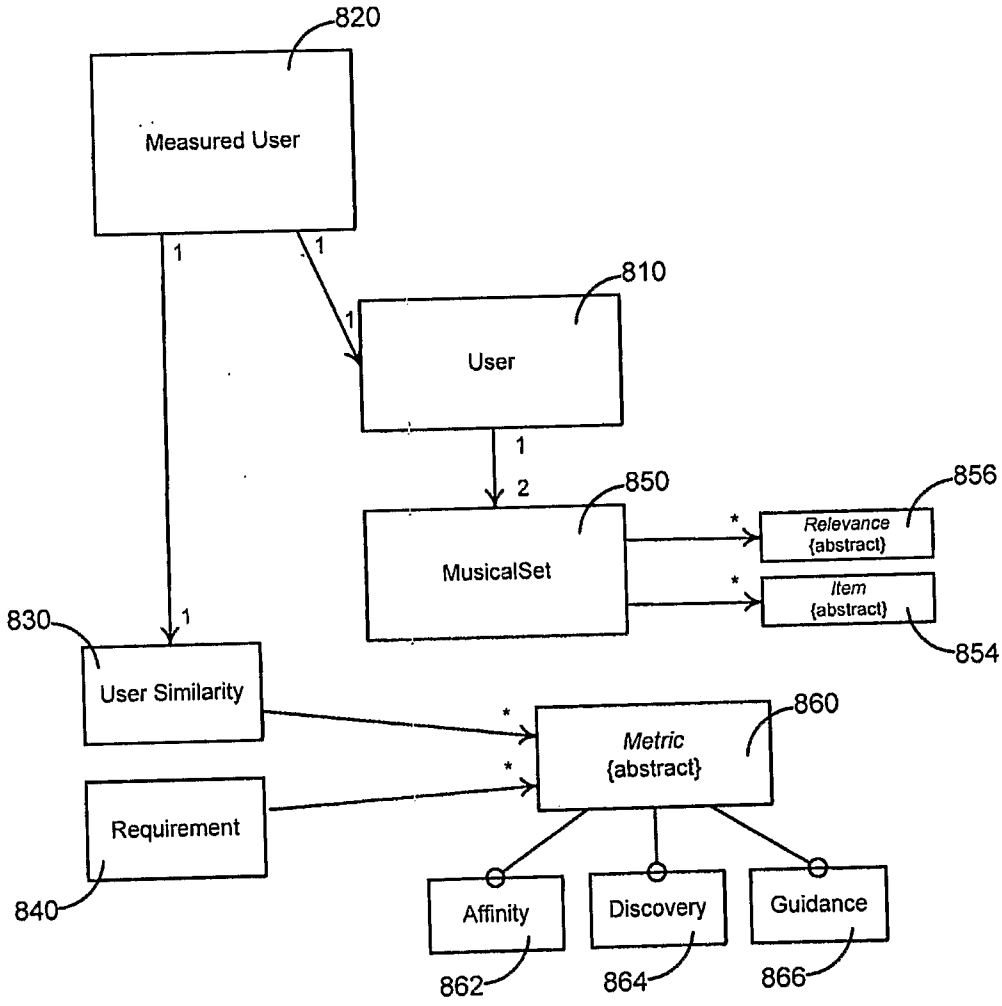

820
Measured User
1
1
1
810
User
1
2
850
MusicalSet
*
Relevance
{abstract}
856
*
Item
{abstract}
854
830
1
User Similarity
*
*
Metric
{abstract}
860
Requirement
840
Affinity
Discovery
Guidance
862
864
866

FIG. 8

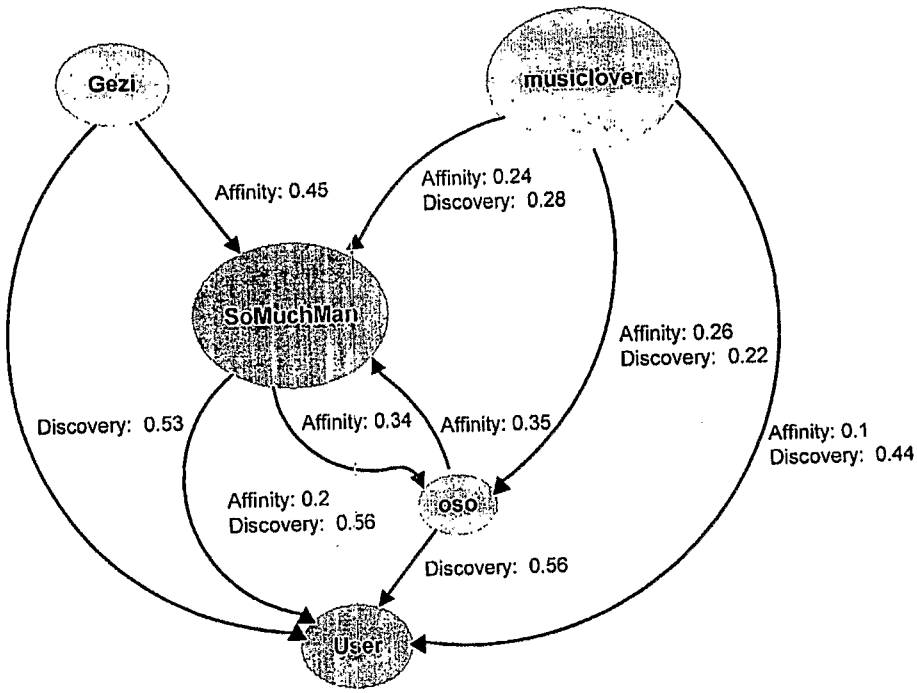

Gezi
musiclover
Affinity: 0.45
Affinity: 0.24
Discovery: 0.28
SoMuchMan
Affinity: 0.26
Discovery: 0.22
Discovery: 0.53
Affinity: 0.34
Affinity: 0.35
Affinity: 0.1
Discovery: 0.44
Affinity: 0.2
Discovery: 0.56
oso
Discovery: 0.56
User

FIG. 9

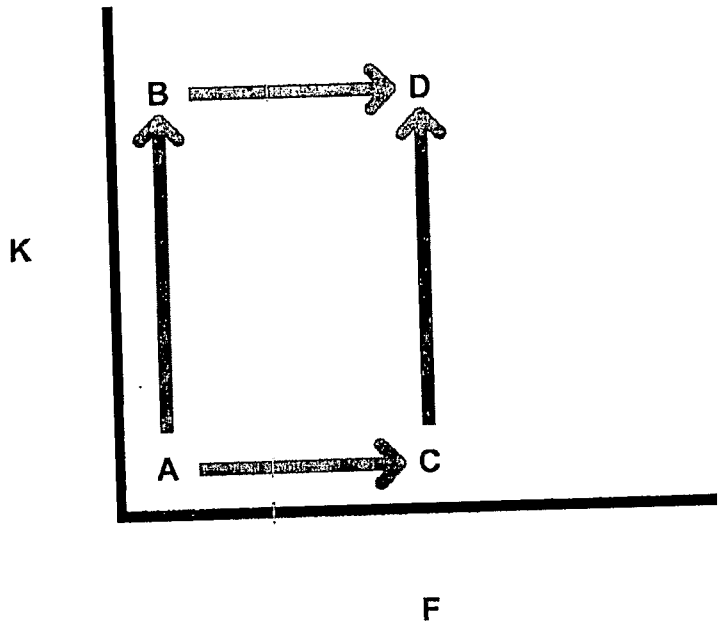


FIG. 10