RECOMMENDATION SYSTEM WITH MULTIPLE INTEGRATED RECOMMENDERS

BACKGROUND

Description of the Related Technology

[0001] Web sites and other types of interactive systems commonly include recommendation systems for providing personalized recommendations of items stored or represented in a data repository. The recommendations are typically generated based on monitored user activities or behaviors, such as item purchases, item viewing events, item rentals, and/or other types of item selection actions. In some systems, the recommendations are additionally or alternatively based on users' explicit ratings of items.

[0002] Traditional collaborative recommendations processes operate by attempting to match users to other users having similar behaviors or interests. For example, once Users A and B have been matched, items favorably sampled by User A but not yet sampled by User B may be recommended to User B. In contrast, content-based recommendation systems seek to identify items having content (e.g., text) that is similar to the content of items selected by the user.

[0003] Other recommendation systems use item-to-item similarity mappings to generate the personalized recommendations. The item-to-item mappings may be generated periodically based on computer-detected correlations between the item purchases, item viewing events, or other types of item selection actions of a population of users. Once generated, a dataset of item-to-item mappings may be used to identify and recommend items similar to those already "known" to be of interest to the user.

SUMMARY

[0004] A recommendations system is disclosed for selecting items to recommend to users. The system comprises a recommendation engine comprising a plurality of recommenders. Each recommender corresponds to a different type of reason for recommending items, and is operative to: retrieve item preference data reflective of actions performed by a user; generate candidate recommendations responsive to a subset of the item preference data, identify one or more reasons for recommending the candidate

recommendations, and score the candidate recommendations to provide relative indications of the strength of the candidate recommendations. The recommendations system also comprises a normalization engine operative to normalize the scores of the candidate recommendations provided by each recommender. The recommendations system further comprises a candidate selector component operative to: select at least a portion of the candidate recommendations based on the normalized scores to provide as recommendations to the user, and output the recommendations with associated reasons for recommending the items.

[0005] Also disclosed is a computer-implemented method of selecting items to recommend. The method comprises: retrieving item preference data reflective of actions performed by a user; and providing the item preference data to a plurality of recommenders, each recommender corresponding to a different type of reason for recommending items. Each recommender is operative to generate candidate recommendations responsive to a subset of the item preference data, and to identify one or more reasons for recommending the candidate recommendations. The method also comprises selecting at least a portion of the candidate recommendations to provide as recommendations to the user; and outputting the recommendations with associated reasons for recommending the items.

[0006] Also disclosed is an apparatus for selecting items to recommend to a user. The apparatus comprises means for retrieving item preference data reflective of actions performed by a user; and means for providing the item preference data to a plurality of recommenders. Each recommender corresponds to a different type of reason for recommending items, and is operative to: generate candidate recommendations responsive to a subset of the item preference data, and identify one or more reasons for recommending the candidate recommendations. The apparatus further includes means for selecting at least a portion of the candidate recommendations to provide as recommendations to the user.

[0007] Also disclosed is a computer-implemented method of normalizing item recommendation scores. The method comprises receiving scores for candidate recommendations from first and second recommenders configured to provide recommendations to a target user, the first recommender operative to assign the scores to the candidate recommendations using a different scoring scale from the second recommender.

The method also comprises, for each recommender, normalizing the scores assigned by the recommender by: calculating a range of scores, the range comprising a difference between a minimum score and a maximum score, and calculating normalized scores as a function of the range. The method further comprises using the normalized scores to select at least a portion of the candidate recommendations to recommend to the target user.

[0008] Also disclosed is computer-implemented method of normalizing item recommendation scores comprising: receiving scores for candidate recommendations from first and second recommenders configured to provide recommendations to a target user, the first recommender configured to assign the scores to the candidate recommendations using a different scoring scale from the second recommender; for each recommender, normalizing the scores assigned by the recommender by: combining the scores for at least some of the candidate recommendations to generate a combined score, and calculating normalized scores as a function of the combined score and the scores for at least some of the candidate recommendations; and using the normalized scores to select at least a portion of the candidate recommendations to recommend to the target user.

[0009] Yet another disclosed computer-implemented method of normalizing item recommendation scores comprises: receiving scores for candidate recommendations from first and second recommenders configured to provide recommendations to a target user, the first recommender operative to assign the scores to the candidate recommendations using a different scoring scale from the second recommender; for each recommender, normalizing the scores assigned by the recommender by assigning percentile rankings to the scores and using the percentile rankings as normalized scores; and using the normalized scores to select at least a portion of the candidate recommendations to recommend to the target user.

[0010] Also disclosed is a system for normalizing item recommendation scores. The system comprises a plurality of recommenders operative to assign scores to candidate recommendations using different scoring scales; and a normalization engine operative to normalize scores assigned by the plurality of recommenders. The normalization engine is operative to: calculate a range of scores, the range comprising a difference between a minimum score and a maximum score, and calculate normalized scores as a function of the

range. The system also comprises a candidate selector operative to use the normalized scores to select at least a portion of the candidate recommendations to recommend to a target user.

[0011] Another system for normalizing item recommendation scores is disclosed that comprises: a plurality of recommenders operative to assign scores to candidate recommendations using different scoring scales; and a normalization engine operative to normalize scores assigned by the plurality of recommenders. The normalization engine is operative to combine the scores for at least some of the candidate recommendations to generate a combined score, and to calculate normalized scores as a function of the combined score and of the scores for at least some of the candidate recommendations. The system also comprises a candidate selector operative to use the normalized scores to select at least a portion of the candidate recommendations to recommend to a target user.

[0012] Another system for normalizing item recommendation scores comprises: a plurality of recommenders operative to assign scores to candidate recommendations using different scoring scales; and a normalization engine operative to normalize scores assigned by the plurality of recommenders. The normalization engine is configured to assign percentile rankings to the scores and to use the percentile rankings as normalized scores. The system also comprises a candidate selector operative to use the normalized scores to select at least a portion of the candidate recommendations to recommend to a target user.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] Specific embodiments will now be described with reference to the drawings, which are intended to illustrate and not limit the various features of the inventions.

[0014] FIGURE 1 illustrates an embodiment of a recommendation system;

[0015] FIGURE 2 illustrates an embodiment of a process for generating item recommendations for a user;

[0016] FIGURE 3A illustrates an embodiment of a process for generating tag-based item recommendations for a user;

[0017] FIGURE 3B illustrates another embodiment of a process for generating tag-based item recommendations for a user;

[0018] FIGURE 4 illustrates an embodiment of a process for normalizing item recommendation scores;

[0019] FIGURE 5 illustrates another embodiment of a process for normalizing item recommendation scores;

[0020] FIGURE 6 illustrates yet another embodiment of a process for normalizing item recommendation scores;

[0021] FIGURE 7 illustrates yet another embodiment of a process for normalizing item recommendation scores;

[0022] FIGURE 8 illustrates a portion of a web page showing an example recommendation interface; and

[0023] FIGURE 9 illustrates one example of how the various recommendation features may be implemented in the context of a web-based electronic catalog system.

DETAILED DESCRIPTION OF SOME EMBODIMENTS

[0024] Several different computer-implemented processes will now be described for using recommendation algorithms to improve item recommendations provided to users. These processes may be embodied individually or in any combination in a multi-user computer system ("system") that implements a recommendation system.

[0025] For purposes of illustration, the processes are described primarily in the context of a system that recommends catalog items to users of an e-commerce web site that provides functionality for users to browse and make purchases from an electronic catalog of items. As will be apparent, however, the disclosed processes can also be used in other types of systems, and can be used to recommend other types of items, such as but not limited to web sites, news articles, blogs, podcasts, travel destinations, service providers, other users, events, discussion boards, photos and other images, videos, tagged items, and user-generated lists of items. In addition, the disclosed processes need not be implemented as part of, or in conjunction with, a web site.

[0026] A significant deficiency in existing recommendation systems, including but not limited to recommendation systems of the types described above, is that they typically use a single, monolithic algorithm for generating recommendations. These algorithms are

often inflexible and not easily adapted to producing recommendations targeted at different customer wants or needs. For example, a recommendation algorithm might recommend items because they are similar to an item the customer purchased. However, on a given day the customer might be interested in shopping for a friend's birthday or exploring new interests rather than buying items similar to what the customer already owns.

[0027] Currently available recommendation systems also fail to provide reasons for recommending items. Customers may not trust the recommendations provided by these systems because they may think that the recommendations are generated by a computer or marketing department that does not understand the customers' needs. This lack of trust can cause customers to ignore their recommendations.

[0028] Turning to **FIGURE 1**, an embodiment of a recommendation system 100 is shown that addresses the foregoing problems, among others. The recommendation system 100 includes multiple recommenders 112 for generating recommendations that target users' varied interests. The recommenders 112 provide reasons for recommending items that can be more compelling than reasons provided by other systems, thereby increasing consumer confidence in the recommendations.

[0029] The various components of the recommendation system 100 may be implemented as software applications, modules, or components on one or more computers, such as servers. While the various components are illustrated separately, they may share some or all of the same underlying logic or code.

[0030] The recommendation system 100 receives item preference data 102 and uses the item preference data 102 to produce personalized item recommendations for a target user. In an embodiment, the item preference data 102 is reflective of actions performed by the user. These actions might include, for example, purchasing items, rating items, adding items to the user's wish list, providing data on the user's friends, tagging items, searching for items, and the like. The item preference data 102 may include browse history data, purchase history data, friends data, tags data, and many other types of data. Some forms of item preference data 102 and their uses will be described more fully below.

[0031] The item preference data 102 is provided to a recommendation engine 110. The recommendation engine 110 includes multiple recommenders 112. In an embodiment,

each recommender 112 may be implemented as a component or algorithm that generates personalized item recommendations targeted to a different interest or need of a user. The multiple recommenders 112 of the recommendation engine 110 can provide more effective recommendations than the monolithic algorithms of currently-available systems.

[0032] In an embodiment, each recommender 112 analyzes a subset of the item preference data to identify items as candidate recommendations for recommending to a user. Each recommender 112 also identifies one or more reasons for recommending the items. As discussed below, different recommenders 112 may use different types of item preference data than others to select candidate items to recommend. Different recommenders 112 may also provide different types of reasons for recommending items.

[0033] For example, a particular recommender 112 might retrieve the user's purchase history data. Using this data, the recommender 112 can find items owned by the user that are part of a series. A series might include, for instance, books in a trilogy, movies and their sequels, or all albums by a musician. If the user has purchased fewer than all the items in the series, the recommender 112 might select the remaining items as candidate recommendations and provide a reason such as, "this item is recommended because you purchased items A and B, and this item would complete your series." Advantageously, this reason can be more compelling than a reason such as "because you purchased items A and B, and this item is similar." Users may therefore be more inclined to trust the reasons provided by the recommenders 112.

[0034] As another example, a recommender 112 might obtain data about a user's friends. This friends data might include information on the friends' birthdays, their wish lists, and their purchase histories. Using this data, a recommender 112 might suggest gifts that could be bought for a friend's upcoming birthday and provide a reason such as "this item is recommended because your friend John's birthday is on July 5th, and this item is on his wish list." Provided with such a reason, the user might be more inclined to buy the item.

[0035] Many other examples of item preference data 102 may be used by the recommenders 112 to generate candidate recommendations and corresponding reasons. For instance, browse history data (e.g., data on user searches, clicks, and the like) may be used to provide a recommendation with the reason, "because this item is similar to an item you

searched for." Purchase history data and/or wish list data might be used to provide a recommendation with the reason, "because this item might be interesting to an early adopter such as you." Browse history data on a browse node of interest to the user (e.g., a category browsed by the user) might be used to provide a recommendation with the reason, "because this item is a top seller in one of your favorite interest areas." Various other forms of item preference data 102 may be used to provide recommendations with reasons such as "because you recently moved," "because you bought an item that may need replacing," "because most people upgrade their DVD player after two years," or the like.

[0036] Multiple reasons may be provided by a single recommender 112, or multiple recommenders 112 may each provide the same candidate recommendation along with a different reason for that recommendation. For instance, several recommenders 112 may be used to recommend a particular war movie because 1) a user recently rated several war movies, 2) this is the best selling movie in the war movie category, and 3) this movie was nominated for two Academy Awards. Using multiple reasons may provide further motivation to the user to view or buy an item.

[0037] However, in certain embodiments, fewer reasons are shown to the user even when multiple reasons are available, to reduce possible information overload. In the above war movie example, the user might therefore only see the reason "because this is the best selling movie in the war movie category." This reason is focused and highly targeted to the user's interest of buying war movies and may be more effective than the multiple reasons provided above.

[0038] The user may also see greater diversity in the reasons that are provided. For example, the user may see one recommendation that is based on an item the user purchased, another based on one or more search queries submitted by the user, and another based on an item listed on a friend's wish list. The diversity of recommendations and reasons provided to the user may heighten user interest in the recommendations.

[0039] Advantageously, in one implementation, at least some of the recommenders 112 are modular. Recommenders 112 can therefore be selectively added to or removed from the recommendation engine 110. As more diverse items or services are added to an online catalog, for instance, new recommenders 112 can be added that target different

user interests. Conversely, some recommenders 112 may be removed from the recommendation engine 110 if they become less useful.

[0040] Some of the recommenders 112 may use particular types of behavior-based associations to select candidate items to recommend. As one example, one recommender may use purchase-based item associations, as generated by mining the purchase histories of large numbers of users, to select candidate items similar to those purchased or owned by the target user. As another example, a particular recommender may use item-viewing based associations, as generated by mining the item viewing histories of large numbers of users, to select candidate items similar to those recently viewed by the target user. Another recommender may use behavior-based associations between particular search queries and items to select candidate items that are related to the search history of the target user. Other recommenders may select candidate items that are unusually popular in the particular geographic region of the target user, or that are unusually popular among users whose email addresses contain the same domain name (e.g., nasa.gov) as the target user. Examples of recommendation methods that use these approaches are described in the following U.S. patent documents, the disclosures of which are hereby incorporated by reference in their entirety: U.S. Patent Nos. 6,853,982 and 6,963,850, and U.S. Appl. No. 10/966,827, filed October 15, 2004. In addition, because the recommenders 112 are modular, the recommenders 112 can be added to an existing recommendation system to improve the quality of recommendations provided by the system.

[0041] The recommenders 112 in certain implementations score the candidate recommendations. The scores can provide indications of the relative strength of the candidate recommendations. Each recommender uses one or more factors to generate the scores. As one example, a recommender 112 that provides recommendations to complete series of items owned by the user might base scores on the total number of items in a series, the number of those items owned by the user, and the sales rank of the items not owned by the user.

[0042] One or more of the recommenders 112 may further take into account negative feedback provided by a user when generating and scoring candidate recommendations, as described in related U.S. Patent Application No. 11/752,251, filed May

22, 2007, and titled "Probabilistic Recommendation System," the disclosure of which is hereby incorporated by reference in its entirety. Negative feedback may be used for items the user has explicitly rated poorly, such as by designating as "not interested" or by rating two stars or less on a scale of 1-5 stars (see FIGURE 7). Other types of negative feedback, including implicit negative feedback, may be used to score candidate recommendations. In an embodiment, negative feedback can cause a candidate recommendation to receive a negative score. A candidate recommendation may also have an overall score that is the sum of both positive scores and negative scores.

[0043] The scores provided by each recommender 112 may be based on factors that might be pertinent to one recommender 112 but not another. For instance, recommendations for top sellers in a browse node of interest to the user might score items based on their relative sales ranking. However, relative sales ranking might not be relevant to recommendations for items an early adopter might buy since there may be little sales data for these items.

[0044] Since the candidate recommendations from each recommender 112 may be based on different criteria, the resulting scores from each recommender 112 can have different scoring scales. One recommender 112 might output, for example, scores in a range of -10,000 to 10,000, whereas another recommender 112 might output scores in a range of 90 to 120. It can be difficult to compare scores from these different score ranges. Moreover, even if the score ranges of different recommenders 112 overlap, the same score outputted by different recommenders may have different meanings because the underlying scoring methodologies may be different. For instance, a score of "2" from one recommender that has a scoring scale of 0 to 100 may have a different meaning than a score of "2" from a recommender that has a scoring scale of 1 to 5.

[0045] Accordingly, in certain embodiments a normalization engine 120 normalizes the scores from the various recommenders 112 to produce normalized scores. The normalized scores enable the candidate recommendations generated by each recommender 112 to be more easily compared. Many different algorithms may be used to normalize the scores. A few example embodiments of these algorithms are described below, with respect to FIGURES 4 through 6.

[0046] Advantageously, the normalization engine 120 facilitates adding or removing modular recommenders 112 to the recommendation engine 110. The normalization engine 120 facilitates this by normalizing scores from any recommender 112 added to the recommendation engine 110. Consequently, recommenders 112 may be added that use different scoring scales from the other recommenders 112. In addition, the normalization engine 130 facilitates removing recommenders 112 from the recommendation engine 110 because scores from the remaining recommenders 112 are normalized and can therefore still be compared.

[0047] In addition to normalizing the scores, the normalization engine 120 can also apply weights to the output from each recommender 112. The weights in one embodiment are multipliers that effectively increase or decrease candidate recommendations' normalized scores. Weights may be applied to emphasize the output of certain recommenders 112 over others. Because some recommenders 112 may produce stronger recommendations than others, applying weights emphasizes the stronger recommendations and deemphasizes the weaker recommendations. Moreover, in some embodiments, the weights may be adjusted for each user to reflect the user's preferences. For instance, if a particular user demonstrates an affinity for items selected by a particular recommender, that recommender's selections may be weighted more heavily for this particular user. These weights may also be adjusted over time to reflect the user's changing interests.

[0048] In some cases, multiple recommenders 112 will generate the same candidate recommendation. One option in this scenario is to add the scores for the candidate provided by each recommender 112. Adding the scores causes a candidate to appear stronger, indicating that candidates provided by multiple recommenders may be good candidates. A potential problem with this approach is that when two recommenders 112 generate a poor candidate, the addition of the scores makes the candidate look stronger than it should. The normalization engine 120 in one embodiment therefore applies exponential decay to the scores, such that scores for the same item are given exponentially less weight as more recommenders 112 recommend the same item. Other decay functions may also be used, such as linear decay.

[0049] The normalization engine 120 passes the candidate recommendations to the candidate selector 130. The candidate selector 130 selects a subset of the candidate recommendations to recommend to the user based on the candidates' normalized scores. For example, the candidate selector 130 may select the N most highly scored candidates to recommend. Alternatively, the candidate selector 130 may select a different subset. For example, in some cases it can be beneficial to show recommendations that are not determined to be the best in order to provide fresh recommendations to the user, among other reasons.

[0050] Rather than selecting a subset of the candidate recommendations, the candidate selector 130 may provide the entire set of candidates to the user. Because this set is typically large (e.g., several thousand items), a user interface used to display the recommendations may allow the user to page or scroll through this recommendations set from highest to lowest ranking. Because users commonly do not take the time to scroll or page through the entire set of recommendations, the practical effect is the same as selecting a subset, e.g., the user is only presented with those items falling near the top of the list.

[0051] In addition, the candidate selector 130 may output, with the recommendations, associated reasons for recommending the items. As described above, a single reason may be provided for each recommendation, or multiple reasons may be provided.

[0052] **FIGURE 2** illustrates an embodiment of a process 200 for generating item recommendations for a user. The process 200 is implemented in one embodiment by a recommendation system, such as the recommendation system 100 of FIGURE 1.

[0053] The process 200 begins at 202 by retrieving item preference data associated with a user. This step may be performed by a recommendation engine, such as the recommendation engine 110 of FIGURE 1. At 204, the process 200 generates candidate recommendations using multiple recommenders. In an embodiment, this step is performed by analyzing item preference data to identify one or more reasons for recommending candidate recommendations to a user.

[0054] At 206, the process 200 scores the candidate recommendations. This step may also be performed by the recommenders. The scores can provide indications of the relative strength of the candidate recommendations. The process 200 in one embodiment

scores candidate recommendations from different recommenders using scoring scales that may be based on factors pertinent to one recommender but not another. In an embodiment, the process 200 also provides negative feedback scores.

[0055] At 208, the process 200 normalizes scores from each recommender. This step may be performed by a normalization engine, such as the normalization engine 120 of FIGURE 1. The normalized scores enable the candidate recommendations to be more easily compared. In an embodiment, this step further includes the step of assigning weights to the scores provided by the recommenders so that some recommenders may be emphasized over others. Additionally, the process 200 may also normalize scores using an exponential decay function, to reduce the effect of the same item being recommended by multiple recommenders.

[0056] At 210, the process 200 selects candidates based on the normalized scores. This step may be performed by a candidate selector, such as the candidate selector 130 of FIGURE 1. The process 200 may select a subset of most highly scored candidates to recommend, or alternatively, provide a different subset of the entire set of candidates as recommendations. At 212, the process 200 outputs recommendations with reasons for recommending the candidate items. This step may also be performed by a candidate selector.

[0057] **FIGURE 3A** illustrates an embodiment of a process 300A for generating tag-based item recommendations for a user. The process 300A is implemented in one embodiment by a recommender, such as one of the recommenders 112 of FIGURE 1.

[0058] Items are tagged in certain embodiments through a user interface that allows users to flexibly apply user-defined tags to individual items in an electronic catalog. The tags may, for example, be in the form of textual annotations or labels that are typed in by users, although other forms of content are possible. The tags and tag-item assignments created by each user are stored persistently in association with the user, and may be kept private to the user or exposed to others. By tagging multiple items with the same tag, a user can flexibly define personal item categories or groupings. For example, a user might create the tag "work" for tagging items relevant to the user's profession, or might create the tag "Tom" for tagging potential items to purchase for a friend or family member named Tom. The users may also have the option to make their tags "public," meaning that these tags are

exposed to other users. Further details on how tags are created are described in U.S. Patent Application No. 11/281,886, filed November 17, 2005, and titled "Recommendations Based on Item Tagging Activities of Users," the disclosure of which is hereby incorporated by reference in its entirety.

[0059] The process 300A begins at 302 by identifying a tagged item associated with, although not necessarily tagged by, a target user. This step is performed in one embodiment by searching item preference data of the target user to find tagged items that the user has purchased, added to a wish list or shopping cart, rated, searched for, or the like. The tags associated with the tagged items need not have been created by the user, although they may have been in some instances. In one embodiment, only public tags are used.

[0060] At 304, the process 300A selects one or more of the tags associated with the tagged item. As items can have multiple tags, the process 300A may select the most popular tag, which may be a tag most frequently attached to the item. Alternatively, the process 300A may select other tags, such as the top three most popular tags.

[0061] Thereafter, the process 300A at 306 performs a search using one or more of the selected tags. The search results are related to the information contained in the tags. Since the tags describe a product associated with the user, at least some of the search results may include items that the user would find interesting. At 308, the process 300A uses at least some of the items in the search result list as candidate recommendations. The process 300A might score the items, for instance, based on search result relevance scores returned by the search engine. In addition, the process 300A may also provide reasons for recommending the items.

[0062] As an example, a user might have purchased a movie in the past starring the fictional character James Bond™. The process 300A can select this movie from the item preference data of the user and determine what tags, if any, are associated with the item. Some possible tags might be "James Bond" and "adventure." The process 300A may then perform a keyword search of an electronic database or catalog using these tags as keywords. The scope of this search may optionally be limited to a particular type of item or collection of items, such as "all products" or "all movies." The search results might include more James Bond™ movies, James Bond™ books, other action or adventure movies, and so forth. Since

at least some of these items are probably related to the movie purchased by the user, some or all of these items may be used as recommendations. Additionally, the process 300A may provide a reason for recommending the items that includes a reference to the tag searched on, such as "recommended because you purchased a movie starring James Bond."

[0063] **FIGURE 3B** illustrates another embodiment of a process 300B for generating tag-based item recommendations for a user. The process 300B is also implemented in one embodiment by a recommender, such as one of the recommenders 112 of FIGURE 1.

[0064] The process 300B begins by identifying a number N of tagged items associated with a target user at 320. As described above, the items may be associated with the user through the user's purchases, items added to a wish list or shopping cart, items the user rated, items the user searched for, or the like. In an embodiment, the process 300B identifies all of the items associated with a user. Alternatively, the process 300B identifies a subset of these items, such as items that were more recently associated with the user. By identifying a subset of the items associated with the user in certain implementations, the process 300B can reduce the processing burden on a system implementing the process 300B.

[0065] At 322, the process 300B identifies tags associated with the N items. Since each item may have multiple tags, there may be a large number of tags among the N items. From this group of tags, the process 300B selects tags at 324 that satisfy specified criteria. For instance, the process 300B might select a threshold number of the most popular tags, such as the ten most popular tags. Or the process 300B might select all tags that were applied to an item a threshold number of times, such as 3 times.

[0066] For each tag, the process 300B then performs a search to obtain a list of scored items at 326. The process 300B does this in one embodiment by sending the tags to a search engine, which performs a search for each tag separately. The search engine returns a ranked list of scored items for each tag searched on. The scores may be based on, for example, the search result scores for each item. In an alternative embodiment, the process 300B performs a search for all of the tags at once, using a logical OR operation.

[0067] The process 300B at 328 merges the lists of scored items while adding scores of alike items. Merging the lists of scored items includes re-ranking the scored items

according to their search results scores to produce a single ranked list of items. The scores of alike items (items appearing in multiple lists) are added to increase the scores, and hence rankings, of these items.

[0068] At 330, the process 300B selects a set of top scored items from the merged list to provide as candidate recommendations. This step can include, for instance, selecting a threshold number of items, such as 10 items, or selecting items having a score above a threshold score.

[0069] **FIGURE 4** illustrates an embodiment of a process 400 for normalizing item recommendation scores. The process 400 is implemented in one embodiment by a normalization engine, such as the normalization engine 120 of FIGURE 1.

[0070] The process 400 begins at 402 by receiving candidate recommendation scores from a recommender. As described above, the scores received from one recommender may differ in scale from scores received from other recommenders. At 404, the process 400 calculates the range of the scores by subtracting the minimum score from the maximum score. Thus, for example, if the minimum score assigned to a candidate recommendation is 10, and the maximum score is 120, then the range is 120 – 10, or 110.

[0071] Thereafter, the process 400 at 406 subtracts the minimum score value from each score provided by the recommender in order to generate a set of translated scores. This step causes the normalized scores to be less than or equal to 1 after step 408. In some embodiments, this step is optional. At 408, the process 400 divides the translated scores by the range to produce normalized scores. The resulting set of normalized scores in one embodiment ranges from 0 to 1.

[0072] The process 400 can be illustrated by an example. Two sets of scores from different recommenders might be as follows: a first set of 1, 3, 5, 2 and a second set of 60, 40, 20, and 10. The range of scores in the first set is 5 – 1 = 4, and the range of scores in the second set is 60 – 10 = 50. Each score in the sets can be translated by the minimum score to generate translated scores: 1 – 1 = 0, 3 – 1 = 2, 5 – 1 = 4, and 2 – 1 = 1 (the new set including 0, 2, 4, and 1); 60 – 10 = 50, 40 – 10 = 30, 20 – 10 = 10, and 10 – 10 = 0 (the new set including 50, 30, 10, and 0). The score sets are then divided by the ranges 4 and 50, respectively, to generate normalized scores 0, 0.5, 1, and 0.25 for the first set and 1, 0.6, 0.2,

and 0 for the second set. Since the scores from each set lie in the same range, they may be compared. Thus, for example, a candidate selector that chooses the top three items from these score sets would choose the item in the first set having score 1 and the items in the second set having scores 1 and 0.6, assuming that the scores from each set are weighted equally.

[0073] As described above, negative scores may be provided by recommenders. The process 400 can also normalize these negative scores. However, when both positive and negative scores are normalized together according to the process 400, the normalized negative scores may be in the range of 0 to 1. Negative scores may therefore have positive normalized scores, eliminating the benefit of adding negative scores to positive scores. In some embodiments, the process 400 overcomes this problem by analyzing negative and positive scores separately. The normalized negative scores can then be subtracted from the positive scores.

[0074] In some embodiments, the process 400 normalizes scores dynamically. In one such implementation, which may be used when the scores are normalized in real time or near real-time, the process 400 normalizes the scores using a window, which may be a list or the like. The window might include, for example, a list of 10,000 scores. In one embodiment, the number of scores in the window increases until a maximum number of scores are reached, such as 10,000 scores. Once the maximum number of scores are reached, the window is reset (e.g., by removing the old scores), and the window begins again to receive new scores. In another embodiment, once the maximum number of scores is reached, each new score added to the window causes an old score to be removed.

[0075] In some embodiments, the window may not include all of the scores generated by a particular recommender. The minimum and maximum scores provided by the recommender may therefore not be in the window. Accordingly, in certain embodiments, the minimum and maximum scores are generated dynamically as the scores are received into the window.

[0076] In one embodiment, the minimum and maximum scores are generated dynamically by determining if a new score inserted into the window is less than a previous minimum score or greater than a previous maximum score. If either of these conditions hold,

then the new score is considered to be the new minimum or maximum. An initial guess of the minimum and maximum scores may be provided when the window is first generated or reset. To conserve processing resources, in one embodiment the minimum and maximum are not evaluated for each new score received by the process 400. Instead, the scores are sampled periodically or probabilistically to evaluate for a new minimum or maximum score. Thus, for example, every 100th score may be evaluated to determine if it is a new maximum or minimum. As the number of scores received in the window increase over time, in some embodiments the minimum and maximum scores stabilize or converge. In certain embodiments, if the window is reset, the calculation of minimum and maximum scores restarts.

[0077] Recommendation scores may be normalized over multiple computers, servers, processors, processing cores, or the like (collectively, "computing devices") to balance processing loads. However, when windowing techniques are used, differences in normalization can occur among the different computing devices. For example, if the same recommender on different computing devices provides different scores to a normalization engine, the minimums and maximums on these computing devices might be calculated differently. The resulting normalized scores might be inconsistent across the different computing devices.

[0078] This inconsistency can undesirably cause different recommendations to be displayed to the same user at different times. Refreshing a web page of recommendations, for instance, can cause a different computing device to generate the recommendations in some embodiments. If the normalization scores are different on each computing device, the refreshed recommendations might be different from the previously-displayed recommendations. These different recommendations may create user confusion and cause user mistrust in the recommendations.

[0079] To reduce the effects of this problem, fewer digits of precision may be used to generate the scores, or alternatively, the process 400 may reduce the number of digits of precision in each score. In effect, the process 400 selects a subset of digits used in the scores. Thus, a score of 0.529 might be modified to become simply 0.5. Advantageously, with fewer digits of precision in the scores, the faster the minimums and maximums can

stabilize or converge. In effect, variability among the normalized scores on the different computing devices is reduced because common ranges are found more quickly on each server.

[0080] Sometimes outliers in a set of scores can skew the distribution of normalized scores. Outliers include scores that are much smaller or much larger than most of the other scores. For example, in a set of scores 1, 2, 5, and 1001, the score 1,001 might be an outlier. Outliers can skew the normalized distribution by affecting the range. In the above example, the range is 1000. Dividing the various scores by this number (after translation by the minimum value) yields normalized scores 0, 0.001, 0.004 and 1. The outlier in this example overwhelmingly dominates the other normalized scores.

[0081] Outliers may indicate very strong recommendations and therefore may be desirable to keep. However, when outliers overpower the other recommendations (such as in the above example), it may be desirable to discard the outliers. One way of doing this is to have each recommender remove the outliers. For example, a recommender could set a threshold and remove scores above the threshold (or below the threshold, in the case of low-valued outliers). Another way to remove outliers when dynamic normalization is used is to use the window technique described above, periodically resetting the window. For example, instead of using every score or even a sample of every score to generate minimums and maximums, the minimums and maximums could be reset after a certain number of scores (e.g., after 1000 scores) have been normalized. That way, the impact of outliers is lessened because the reset causes old minimums and maximums to be ignored for future calculations. Yet another way of reducing the impact of outliers is taking the Nth largest (or Nth smallest) score as the maximum (or minimum) score. For instance, the second-to-largest score may be chosen as the maximum score instead of the largest score.

[0082] **FIGURE 5** illustrates another embodiment of a process 500 for normalizing item recommendation scores. The process 500 is implemented in one embodiment by a normalization engine, such as the normalization engine 120 of FIGURE 1.

[0083] Like the process 400, the process 500 begins at 502 by receiving candidate recommendation scores from a recommender. At 504, the process 500 determines an original range of the scores. This original range may be determined, for example, by subtracting a

minimum score from a maximum score. This range may be calculated dynamically using the window techniques described above.

[0084] At 506, the process 500 determines a new range. This new range includes a new minimum value and a new maximum value. In one example embodiment, the new range is 0 to 1. Another example range might be -10 to 10. Other ranges may be chosen without limitation.

[0085] At 508, the process 500 maps the scores from the original range to the new range using a mathematical transformation. The transformation in one embodiment is a nonlinear transformation. The transformation in certain embodiments takes the form

$$\text{Normalized Score} = (\text{Item Score} - \text{Old Min})*(\text{New Range}/\text{Old Range}) + \text{New Min}, \quad (1)$$

where the Item Scores are candidate recommendation scores, where Old Range = Old Max – Old Min, and where New Range = New Max – New Min. Expression (1) illustrates that for each Item Score, a new score is generated as a function of the Item Scores, the Old Range, and the New Range.

[0086] It will be noted that when New Min is 0 and New Range is from 0 to 1, expression (1) becomes

$$\text{Normalized Score} = (\text{Item Score} - \text{Old Min})/\text{Old Range}. \quad (2)$$

The normalized scores in expression (2) are computed in the same or a similar manner as the normalized scores of FIG. 4. The minimum value in expression (2) is subtracted from each item score to produce translated scores, which are divided by the range.

[0087] Like the process 400, the process 500 can also use the techniques of the process 400 to calculate negative scores, to increase consistency among normalized scores across multiple computing devices, and to minimize the impact of outliers.

[0088] **FIGURE 6** illustrates another embodiment of a process 600 for normalizing item recommendation scores. The process 600 is implemented in one embodiment by a normalization engine, such as the normalization engine 120 of FIGURE 1.

[0089] The process 600 begins at 602 by receiving candidate recommendation scores from a recommender. At 604, the process 600 determines whether a minimum score from the set of received candidate recommendation scores is different from a desired normalized minimum score. The desired normalized minimum score in one embodiment is the value that will be chosen as the minimum score in the normalized range of scores.

[0090] If the minimum candidate recommendation score differs from the desired minimum normalized score, the process 600 at 606 translates each score in the set of candidate recommendation scores by a difference between the minimum candidate recommendation score and the desired normalized minimum score. As one example, a set of candidate recommendation scores might have a minimum score of 90 on a scale of 90 to 100. If the desired normalized minimum score is 0, the minimum score and the desired normalized minimum score differ by 90. Accordingly, each candidate recommendation score will be translated (e.g., subtracted) by 90, resulting in a new set of scores ranging from a minimum of 0 to a maximum of 10. Translating the candidate recommendation scores advantageously enables sets of scores from different recommenders having different minimum scores to be more easily compared. After translating the scores, the process 600 proceeds to step 608. If, however, the minimum candidate recommendation score is determined to be the same as the desired normalized minimum score at 604, the process 600 proceeds directly to step 608. In addition, it should be noted that in alternative embodiments, steps 604 and 606 may be omitted.

[0091] At 608, the process 600, in certain embodiments, combines the scores of all the items to create a combined score. In one embodiment, combining the scores is done by summing the scores. In another embodiment, block 608 is performed by computing a moving average of a subset of the scores and multiplying the average by the number of scores in the moving average. The moving average may be implemented, for example, by using one or more of the window techniques described above. A moving average can reduce the processing burden on a computing system by reducing the number of calculations to be performed, since the average of all the scores is not computed each time a new score is received. In an embodiment, the moving average is an exponential moving average.

[0092] Thereafter, the process 600 calculates normalized scores by using the combined score and the candidate recommendation scores. This step is performed, for example, by dividing each candidate recommendation score by the combined score. In embodiments where the desired minimum normalized score is 0, the normalized scores might range from 0 to 1.

[0093] The process 600 may use a window technique, such as described above with respect to FIG. 4, to calculate the minimum candidate recommendation score. Advantageously, the process 600 of certain embodiments can also use the techniques of the process 400 to calculate negative scores, to increase consistency among normalized scores across multiple computing devices, and to minimize the impact of outliers. Additionally, in one embodiment the process 600 also reduces the impact of outliers by periodically resetting a window of scores when window techniques are used. Resetting the window removes the impact of previous outliers. Conversely, the number of scores in the window could be allowed to increase (e.g., the window would be reset at longer intervals), spreading out the affect of outliers on the normalized scores.

[0094] **FIGURE 7** illustrates yet another embodiment of a process 700 for normalizing item recommendation scores. The process 700 is implemented in one embodiment by a normalization engine, such as the normalization engine 120 of FIGURE 1.

[0095] The process 700 begins at 702 by receiving candidate recommendation scores from a recommender. Thereafter, the process 700 assigns percentile rankings to the scores. In an embodiment, a score's percentile ranking (or equivalently, a candidate recommendation's percentile ranking) reflects the strength of a particular candidate's score. Thus, for instance, a candidate recommendation in the 95th percentile has a score that is higher than 95% of the other candidates' scores. It will be understood, however, that in other implementations the percentile rankings may be used to determine the weakness of a candidate's score. For example, a candidate recommendation in the 95th percentile in this implementation might have a score that is lower than 95% of the other candidates' scores.

[0096] The percentile rankings may be assigned in a variety of ways. One way is to calculate the mean and variance values of the set of candidate recommendation scores and use these values to derive the percentile rankings from a normal distribution having the

calculated mean and variance. The percentile rankings generated from the normal distribution may be obtained from a lookup table or the like. The process 700 in one embodiment may use a window technique, such as described above with respect to FIG. 4, to calculate the mean and variance values.

[0097] Alternatively, the percentiles may be calculated using the following formula:

$$\text{Percentile} = \text{Rank}/\text{Number of scores.} \qquad (2)$$

The term "Rank" in expression (2) refers to the numerical position of a candidate score in the set of candidate scores. If, for example, there are 100 candidate recommendation scores and a particular score is the 50th highest score, its percentile is 50/100 = 0.5. Expression (2) generates percentile rankings ranging from 0 to 1, but the percentiles may alternatively be expressed as percentages (e.g., "50%" or "50th percentile").

[0098] In an embodiment, the percentile rankings are generated dynamically using a window of scores, using similar techniques to those described above with respect to FIG. 4. In one such embodiment, the window is implemented as a sorted list of scores, where an old score is removed from the list each time a new score is inserted into the list. Since the scores are sorted, a percentile ranking can be derived from each score's position or rank in the list using, for example, expression (2). For example, the first position in the list might be ranked 1st, the second position might be ranked 2nd, and so on. In an embodiment, the list is sorted automatically as new scores are inserted into the list. The position in the list where the new score is inserted can be determined by searching the list to find the correct position for the new score. In one embodiment, the new score replaces an old score in the same position in the list. Alternatively, the oldest score in the list, regardless of position, is removed from the list when the new score is inserted.

[0099] The process 700 at 706 uses the percentile rankings as normalized scores. Advantageously, using percentile rankings as normalized scores reduces the sensitivity of the normalized scores to outliers. As a result, the process 700 may not need to account for unusually low or high scores. Percentile rankings are generally insensitive to outliers because

the rankings of successively-ranked scores tend to be independent of the quantitative difference between those scores. For example, a first set of scores 1, 2, and 3 would be ranked the same way as a second set of scores 1, 2, and 100.

[0100] The process 700 of certain embodiments can also use the techniques of the process 400 to increase consistency among normalized scores across multiple computing devices. In addition, negative scores may be calculated separately from positive scores, as described above. For items that have only negative scores, the percentile rankings can be reversed, such that an item with a very negative score will have a very low percentile ranking.

[0101] **FIGURE 8** illustrates a portion of a web page showing an example recommendation interface. The example recommendations page 800 displays recommendations for a user. The recommendations page 800 includes various details about the listed products 810 (four products shown), and includes buttons for adding each product to an electronic shopping cart or wish list. In this example, the recommendation page 800 also includes a set of controls 812 for rating, indicating ownership of, and indicating a lack of interest in, each listed product 810. The recommendations system may use this information and other information to improve the recommendations it makes. In an embodiment, this process is stateless, such that no information about which items have been recommended to which users needs be retained. A refresh option 818 allows a user to see an updated list of recommendations, which may be updated when the user adjusts the controls 812.

[0102] One or more reasons 814 are displayed for recommending each item. For example, the item "The Arctic Incident" includes the reason 814a "Recommended because you said you owned The Eternity Code (Artemis Fowl, Book 3), and this item will complete your series." As described above, the reasons 814 provide compelling reasons for recommending items. A user that owns one or more books in the Artemis Fowl series, for instance, might be interested in buying the remaining books in the series.

[0103] **FIGURE 9** illustrates a set of components that may be included in an electronic catalog website 986 to implement the recommendation functions described above. The system may also include functionality for users to perform various types of item-related actions such as purchasing items, tagging items, adding items to personal wish lists and shopping carts, rating items, reviewing items, etc. The arrows in FIGURE 9 show the general

flow of information between components. As illustrated by FIGURE 9, the system may be accessed by user computers 988 over the Internet. Although shown as personal computers for purposes of illustration, the user computers 988 may include various other types of computing devices, including Personal Digital Assistants (PDAs), wireless phones, set-top television boxes, etc. In the embodiment shown in FIGURE 9, the system 986 comprises web servers 990 which process HTTP (Hypertext Transfer Protocol) requests received over the Internet from the user computers 988 that run web browser software. In response to user requests, the web servers 990 dynamically generate content-dependent web pages according to user-specific information. The web servers 990 access a repository of web page templates 992 that specify the layout and format of product detail pages, recommendations pages, and various other types of web pages. The web servers 990 populate these templates with information that is typically dependent upon the identity of the particular user, as may be determined, for example, using browser cookies.

[0104] As further illustrated in FIGURE 9, the web servers 990 retrieve catalog content for particular products from a Catalog Service 994, which includes or accesses a repository 996 of item content. The item content may, for example, include photos, reviews, price and availability data, and other types of descriptive information about particular products that are available to purchase, rent, download, review, post for sale, etc. via the web site 986.

[0105] The web servers 990 also communicate with a tagging service 998 that maintains a database 900 of user-specific tag data. The tag data stored for each user may, for example, include a set of tag-item ID pairs, optionally together with various other types of data such as permission data and a creation timestamp. The tagging service 998 may receive both read requests from the web servers (e.g., when a user requests a page that displays personal tag data), and update requests (e.g., when a user tags an item). In one embodiment, each tag is stored in association with the corresponding user, meaning that if two or more users create identical tags, these tags are treated as separate and distinct from each other. The tags may also be stored in association with one or more items in the electronic catalog.

[0106] The web servers 990 also communicate with a search engine 904 that allows users to search for information stored in the item content and tag data repositories 996

and 900. As mentioned above, the search engine may be used to generate recommendations by searching using tags of various items as keywords.

[0107] The web servers 990 also access a recommendations service 901 which generates item recommendations. The recommendation service 901 may include multiple recommenders and a normalization engine as shown in FIGURE 1 and described above. To request recommendations for a particular user, the a web server 990 sends a request to the recommendations service 901, which responds with a list of recommended items according to the systems and processes described above with respect to FIGURES 1-8. The recommendation service 901 may generate the recommendations in real time in response to a particular user action.

[0108] For example, in one embodiment, when a user clicks on a link that invokes the presentation of personalized recommendations, the system generates and returns item recommendations in real time as follows. Initially, a web server 990 sends a request to the recommendation service 901. The recommendation service then responds by invoking some or all of its recommenders 112. The recommenders 112 may, but need not, be invoked in parallel. Each invoked recommender 112 responds by retrieving item preference data 902, which may be distributed over several servers. Each recommender 112 then generates a list of candidate items for the user, together with associated scores and reasons. The normalization engine 120 normalizes the scores as described above, and the candidate selector 130 then uses the normalized scores to select particular candidate items to recommend, and/or to rank the candidate items for display. The recommendation service 901 then returns the ranked list of items and the associated reasons to the web server 990. Finally, the web server 990 uses this information, together with item data retrieved from the catalog service 994 (and possibly other services 906), to generate and return a recommendations page of the type shown in FIGURE 8.

[0109] Reasons are generated in one implementation by providing several predefined reason types that identify different kinds of reason text. A lookup table or the like may be provided, for example, that maps reason types to reason text. For instance, a reason type "A" might map to the reason text "because you purchased item X," and a reason type "B" might map to the reason text "because item X is on your friend's wish list." In one

embodiment, the recommenders 112 pass reason types along with candidate recommendations to the normalizer 120. In turn the normalizer 120 passes the reason types and candidate recommendations to the candidate selector 130, which passes certain recommendations along with their reason types to a user interface component (not shown). The user interface component matches reason types with reason text according to the lookup table and displays the recommendations with the associated reason text to a user (see, e.g., FIG. 8).

[0110] In some cases, a particular recommender 112 may not return any candidate items. This may be the case where, for example, the user has not engaged in a particular type of user activity on which the recommender is based, or where the recommender otherwise relies on some type of user data that is not available for the particular user.

[0111] The recommendations service 901 also communicates with the tagging service in certain embodiments to obtain tagging data useful for producing recommendations, according to the process 300 described with respect to FIG. 3 above. The recommendations service 901 also optionally communicates with one or more other services 906, such as a friends service that allows the user to save birthday and interest data about friends.

[0112] The web servers 990 also access one or more additional repositories of user data, logically represented in FIGURE 9 as item preference data 902. Because a group of individuals can share an account, a given "user" may include multiple individuals (e.g., two family members that share a computer). As illustrated by FIGURE 9, the data stored for each user may include one or more of the following types of information (among other things) that can be used to generate recommendations in accordance with the invention: (a) the user's purchase history, including dates of purchase, (b) a history of items recently viewed by the user, (c) the user's item ratings profile, if any, and (d) items tagged by the user. Various other types of user information, such as wish list/registry contents, email addresses, shipping addresses, shopping cart contents, and browse (e.g., clickstream) histories, may additionally be stored.

[0113] The various components of the web site system 986 may run, for example, on one or more servers (not shown). In one embodiment, various components in or communicating with the recommendations service 901 are replicated across multiple machines to accommodate heavy loads.

[0114] Each of the processes and algorithms described above may be embodied in, and fully automated by, code modules executed by one or more computers or computer processors. The code modules may be stored on any type of computer-readable medium or computer storage device. The processes and algorithms may also be implemented partially or wholly in application-specific circuitry. The results of the disclosed processes and process steps may be stored, persistently or otherwise, in any type of computer storage.

[0115] The various features and processes described above may be used independently of one another, or may be combined in various ways. All possible combinations and sub-combinations are intended to fall within the scope of this disclosure. In addition, certain method or process steps may be omitted in some implementations.

[0116] Although this disclosure has been described in terms of certain example embodiments and applications, other embodiments and applications that are apparent to those of ordinary skill in the art, including embodiments and applications that do not provide all of the benefits described herein, are also within the scope of this disclosure. The scope of the inventions is defined only by the claims, which are intended to be construed without reference to any definitions that may be explicitly or implicitly included in any of the incorporated-by-reference materials.

WHAT IS CLAIMED IS:

1. A recommendations system for selecting items to recommend to a user, the system comprising:

a recommendation engine comprising a plurality of recommenders, each recommender corresponding to a different type of reason for recommending items, each recommender operative to:

retrieve item preference data reflective of actions performed by a user;

generate candidate recommendations responsive to a subset of the item preference data,

identify one or more reasons for recommending the candidate recommendations, and

score the candidate recommendations to provide relative indications of the strength of the candidate recommendations;

a normalization engine operative to normalize the scores of the candidate recommendations provided by each recommender; and

a candidate selector component operative to:

select at least a portion of the candidate recommendations based on the normalized scores to provide as recommendations to the user, and

output the recommendations with associated reasons for recommending the items.

2. The system of Claim 1, wherein at least some of the recommenders are modular, such that the recommenders can be selectively added to or removed from the recommendation engine.

3. The method of Claim 1, wherein the different type of reason for recommending items is that a recommendation could complete a series of items owned by the user.

4. The method of Claim 1, wherein the different type of reason for recommending items is that a recommendation could be used as a gift for a friend's upcoming birthday.

5. A computer-implemented method of selecting items to recommend, the method comprising:

retrieving item preference data reflective of actions performed by a user;

providing the item preference data to a plurality of recommenders, each recommender corresponding to a different type of reason for recommending items, each recommender operative to:

generate candidate recommendations responsive to a subset of the item preference data, and

identify one or more reasons for recommending the candidate recommendations;

selecting at least a portion of the candidate recommendations to provide as recommendations to the user; and

outputting the recommendations with associated reasons for recommending the items.

6. The method of Claim 5, wherein each recommender is further configured to generate scores for the candidate recommendations.

7. The method of Claim 6, further comprising normalizing the scores from each recommender.

8. The method of Claim 5, wherein at least some of the recommenders are modular, such that the recommenders can be selectively added to or removed from a recommendation engine.

9. The method of Claim 5, wherein the different type of reason for recommending items is that a recommendation could complete a series of items owned by the user.

10. The method of Claim 5, wherein the different type of reason for recommending items is that a recommendation could be used as a gift for a friend's upcoming birthday.

11. The method of Claim 5, wherein the different type of reason for recommending items is that a recommendation is similar to an item searched for by the user.

12. The method of Claim 5, wherein the different type of reason for recommending items is that a recommendation is an item that might be purchased by an early adopter, and the user is an early adopter.

13. The method of Claim 5, wherein the different type of reason for recommending items is that a recommendation is a top seller in a browse node of interest to the user.

14. The method of Claim 5, wherein the item preference data comprises data selected from the group consisting of browse history data, series data, search data, purchase history data, friends data, browsing interests data, tags data, and top sellers data.

15. The method of Claim 5, wherein each recommender is further configured to provide one or more reason types that map to reason text.

16. A computer-readable medium having instructions stored thereon which, when executed, cause a computer system to perform the method of Claim 5.

17. A computer system operative to perform the method of Claim 5.

18. An apparatus for selecting items to recommend to a user, the apparatus comprising:

means for retrieving item preference data reflective of actions performed by a user;

means for providing the item preference data to a plurality of recommenders, each recommender corresponding to a different type of reason for recommending items, each recommender configured to:

generate candidate recommendations responsive to a subset of the item preference data, and

identify one or more reasons for recommending the candidate recommendations; and

means for selecting at least a portion of the candidate recommendations to provide as recommendations to the user.

19. The apparatus of Claim 18, further comprising means for outputting the recommendations with associated reasons for recommending the items.

20. A computer-implemented method of normalizing item recommendation scores, the method comprising:

receiving scores for candidate recommendations from first and second recommenders configured to provide recommendations to a target user, the first recommender operative to assign the scores to the candidate recommendations using a different scoring scale from the second recommender;

for each recommender, normalizing the scores assigned by the recommender by:

calculating a range of scores, the range comprising a difference between a minimum score and a maximum score, and

calculating normalized scores as a function of the range; and

using the normalized scores to select at least a portion of the candidate recommendations to recommend to the target user.

21. The method of Claim 20, further comprising translating the scores prior to calculating the scores as a function of the range.

22. The method of Claim 20, wherein calculating normalized scores as a function of the range comprises dividing the scores by the range.

23. The method of Claim 20, wherein normalizing the scores assigned by the recommender further comprises determining a new range for the normalized scores.

24. The method of Claim 23, wherein calculating normalized scores as a function of the range comprises mapping the scores from the range to the new range by applying a transformation to the scores.

25. The method of Claim 20, wherein calculating a range of scores comprises using a window to calculate a new minimum score if a score received from a recommender is less than the minimum score and calculating a new maximum score if a score received from a recommender is greater than the maximum score.

26. The method of Claim 25, further comprising resetting the window.

27. The method of Claim 20, wherein normalizing the scores assigned by the recommender further comprises:

normalizing the scores using a plurality of computing devices; and

reducing a number of digits of precision in the scores, such that variability in the normalized scores among the computing devices is reduced.

28. The method of Claim 20, further comprising:

partitioning the scores into a set of positive scores corresponding to positive candidate recommendations and a set of negative scores corresponding to negative candidate recommendations; and

normalizing the positive scores separately from the negative scores.

29. The method of Claim 28, further comprising combining normalized positive scores with normalized negative scores to obtain overall normalized scores for the candidate recommendations.

30. The method of Claim 20, further comprising applying exponential decay to scores for equivalent items provided by both the first and second recommenders.

31. The method of Claim 20, further comprising assigning weights to candidate recommendations received from the first and second recommenders.

32. The method of Claim 31, further comprising adjusting the weights based on direct or indirect feedback received from the user.

33. A computer-readable medium having instructions stored thereon which, when executed, cause a computer system to perform the method of Claim 20.

34. A computer-implemented method of normalizing item recommendation scores, the method comprising:

receiving scores for candidate recommendations from first and second recommenders configured to provide recommendations to a target user, the first recommender configured to assign the scores to the candidate recommendations using a different scoring scale from the second recommender;

for each recommender, normalizing the scores assigned by the recommender by:

combining the scores for at least some of the candidate recommendations to generate a combined score, and

calculating normalized scores as a function of the combined score and the scores for at least some of the candidate recommendations; and

using the normalized scores to select at least a portion of the candidate recommendations to recommend to the target user.

35. The method of Claim 34, further comprising translating the scores by a difference between a minimum score and a desired normalized minimum score in response to determining that the minimum score differs from the desired normalized minimum score.

36. The method of Claim 34, further comprising adjusting the number of scores combined in the combined score to reduce the impact of outliers on the normalized scores.

37. The method of Claim 34, wherein combining the scores for at least some of the candidate recommendations comprises summing the scores.

38. The method of Claim 34, further comprising assigning weights to candidate recommendations received from the first and second recommenders.

39. A computer-readable medium having instructions stored thereon which, when executed, cause a computer system to perform the method of Claim 34.

40. A computer-implemented method of normalizing item recommendation scores, the method comprising:

receiving scores for candidate recommendations from first and second recommenders configured to provide recommendations to a target user, the first recommender operative to assign the scores to the candidate recommendations using a different scoring scale from the second recommender;

for each recommender, normalizing the scores assigned by the recommender by assigning percentile rankings to the scores and using the percentile rankings as normalized scores; and

using the normalized scores to select at least a portion of the candidate recommendations to recommend to the target user.

41. The method of Claim 40, wherein assigning percentile rankings to the scores comprises dynamically assigning percentile rankings to the scores as the scores are received from the recommenders.

42. The method of Claim 40, further comprising:

partitioning the scores into a set of positive scores corresponding to positive candidate recommendations and a set of negative scores corresponding to negative candidate recommendations; and

normalizing the positive scores separately from the negative scores.

43. The method of Claim 42, further comprising reversing the rankings of the negative scores.

44. A computer-readable medium having instructions stored thereon which, when executed, cause a computer system to perform the method of Claim 40.

45. A system for normalizing item recommendation scores, the system comprising:

a plurality of recommenders operative to assign scores to candidate recommendations using different scoring scales;

a normalization engine operative to normalize scores assigned by the plurality of recommenders, the normalization engine operative to:

calculate a range of scores, the range comprising a difference between a minimum score and a maximum score, and

calculate normalized scores as a function of the range; and

a candidate selector operative to use the normalized scores to select at least a portion of the candidate recommendations to recommend to a target user.

46. A system for normalizing item recommendation scores, the system comprising:

a plurality of recommenders operative to assign scores to candidate recommendations using different scoring scales;

a normalization engine operative to normalize scores assigned by the plurality of recommenders, the normalization engine operative to:

combine the scores for at least some of the candidate recommendations to generate a combined score, and

calculate normalized scores as a function of the combined score and of the scores for at least some of the candidate recommendations; and

a candidate selector operative to use the normalized scores to select at least a portion of the candidate recommendations to recommend to a target user.

47. A system for normalizing item recommendation scores, the system comprising:

a plurality of recommenders operative to assign scores to candidate recommendations using different scoring scales;

a normalization engine operative to normalize scores assigned by the plurality of recommenders, the normalization engine configured to assign percentile rankings to the scores and use the percentile rankings as normalized scores; and

a candidate selector operative to use the normalized scores to select at least a portion of the candidate recommendations to recommend to a target user.