Chapter 2
Recommender Systems

Recommender systems base their operation on past user purchases/ratings over a collection of items, for instance, books, CDs, etc. In this chapter, we bring to surface factors that affect the recommendation algorithmic process. Moreover, we describe the most important problems related to recommender systems and give some references to actual solutions. Finally, there is an economic and social report regarding Recommender Systems, which examines them under a more market-based angle.

2.1 Basic Approaches

Recommender systems are gaining widespread acceptance in e-commerce applications as a way of tackling the "information overload" problem. This problem affects our everyday experience while searching for information on a topic. To overcome this problem, we often rely on suggestions from others who have more experience on the topic. However, in the Web case where there are numerous suggestions, it is not easy to detect the trustworthy ones. The process of recommendation becomes controllable by shifting from individual to collective suggestions,

Three parallel approaches have emerged in the context of recommender systems: collaborative filtering (CF), content-based Filtering (CB) and hybrid methods.

Collaborative filtering algorithms recommend those items to the target user, that have been rated highly by other users with similar preferences and tastes [24, 28]. In most CF approaches, only the item and users' identifiers are accessible and no additional information over items or users is provided. Websites that provide recommendations in the form, "Customers who bought item i also bought item y", typically fall under collaborative filtering approaches. Grouplens research group [24] introduced a collaborative filtering algorithm, known as user-based CF, because it employs users' similarities for the formation of the neighborhood of nearest users. Another CF algorithm proposed by Sarwar et al. [28], is known as

P. Symeonidis et al., *Recommender Systems for Location-based Social Networks*,
SpringerBriefs in Electrical and Computer Engineering,
DOI 10.1007/978-1-4939-0286-6_2, © The Author(s) 2014

item-based CF algorithm, because it employs items' similarities for the formation of the neighborhood of nearest users. A pitfall of CF is the cold start problem: new items have received only few ratings, so they cannot be recommended; new users have performed only few transactions, so other users similar to them can be hardly found.

Content-based filtering assumes that each user operates independently. It exploits only information derived from documents or item features (eg. terms or attributes) [4, 20, 23]. In particular, it exploits a set of attributes, which describes the items and recommends other items similar to those that exist in the user's profile. This way, the cold start problem for new items and new users are alleviated, *provided* that users prefer items that are similar in content to those they have already chosen. However, the pitfall of CB is that there is no diversity in the recommendations. That is, the user gets recommendations that are very familiar to her, since the recommended items are similar to those in her item profile.

Hybrid algorithms attempt to combine CB with CF. The combination of content with rating data helps capture more effective correlations between users or items, which yields more accurate recommendations. The Fab System [1], combines CB and CF in its recommendations, by measuring similarity between users after first computing a profile for each user. Fab initially categorizes documents by a CB filter and then recommends them to the test user based on his relevance feedback. In contrast, the CinemaScreen System [25] runs CB on the results of CF. In particular, the CinemaScreen system computes predicted rating values for movies based on CF and then applies CB to generate the recommendation list.

Finally, apart from blending the content with rating data, Social web has allowed the emergence of new data combinations that can provide even more robust recommendations [22]. For instance, social networks such as Facebook, LinkedIn, etc., include information about the connections (link data) between humans. There are two main types of recommendations in social networks. The first one is related to the link prediction task, whereas the second one refers to the rating prediction and item recommendation task.

2.2 Definitions and Basic Factors

In this section, we identify the major factors that critically affect all CF algorithms, since they are among the most popular methods in recommender systems. Our analysis focuses on the basic operations of the CF process, which consists of three stages.

- *Stage 1*: formation of the user or item neighborhood with objects of similar ratings and behavior.
- *Stage 2*: generation of a top-N list with algorithms that construct a list of best item recommendations for a user.
- *Stage 3*: quality assessment of the top-N list.

Table 2.1 Factors affect CF algorithms

Factor name	Short description	Stage
Sparsity	Limited percentage of rated products	1
Scalability	Computation increase by the number of users and items	1
Train/test data size	Data are divided into training and evaluation or test	1,3
Neighborhood size	Number of neighbors used for the neighborhood formation	1
Similarity measure	Measures that calculate proximity of two objects	1
Recommendation list size	Number of top-N recommended items	2
Recommendation list creation	Algorithms for the top-N list generation	2
Positive rating threshold	Positive and negative ratings segregation	2,3
Evaluation Metrics	Metrics that evaluate the quality of top-N list	3
Setting a Baseline method	A simple method against which performance is compared	3
Past/future items	The segregation between a priori known and unknown items	3

In the rest of this section we elaborate on the aforementioned factors, which are organized with respect to the stage where each one is involved. The examined factors,which are detailed in the following, are succinctly described in Table 2.1.

Table 2.2 summarizes the symbols used in the sequel. To ease the discussion, we will use the running example illustrated in Fig. 2.1, where U_{1-10} are users and I_{1-6} are items. As shown, the example data set is divided into a training and a test set. Null cells (no rating) are represented as zeros.

Note that, in addition, a few more factors have been identified, like the impact of subjectivity during the rating or issues related to the preprocessing of data [3,19,27]. Nevertheless, we do not examine these factors, because their effect is less easily determinable.

2.2.1 First Stage Factors

Sparsity: In most real-world cases, users rate only a very small percentage of items. This causes data sets to become sparse. The problem of sparsity is extensively studied. In such cases, the recommendation engine cannot provide precise proposals, due to lack of sufficient information. A similar problem of CF algorithms is cold-start, as mentioned previously [21].

In a few works [19, 27], there is a preprocessing step that fills missing values. Several other works [11, 15, 28] focus only on very sparse data. Also, related work provides benchmark data sets with different sparsity, e.g., the Jester data

Table 2.2 Symbols and definitions

Symbol	Definition
k	Number of nearest neighbors
N	Size of recommendation list
$NN(u)$	Nearest neighbors of user u
$NN(i)$	Nearest neighbors of item i
P_τ	Threshold for positive ratings
$\mathcal{I}$	Domain of all items
$\mathcal{U}$	Domain of all users
u, v	Some users
i, j	Some items
$\mathcal{I}_u$	Set of items rated by user u
$\mathcal{U}_i$	Set of users rated item i
$r_{u,i}$	The rating of user u on item i
$\overline{r}_u$	Mean rating value for user u
$\overline{r}_i$	Mean rating value for item i
$p_{u,i}$	Predicted rate for user u on item i
$\|T\|$	Size of the test set
c	Number of singular values
A	Original matrix
U	Left singular vectors of A
S	Singular values of A
V'	Right singular vectors of A
A^*	Approximation matrix of A
$\mathbf{u}$	User vector
$\mathbf{u}_{\mathbf{new}}$	Inserted user vector
n	Number of training users
m	Number of items

a

	I_1	I_2	I_3	I_4	I_5	I_6
U_1	4	1	1	4	0	0
U_2	1	4	4	0	0	4
U_3	2	1	4	0	0	4
U_4	1	2	1	1	0	0
U_5	4	1	2	0	1	0
U_6	1	5	4	0	4	0

b

	I_1	I_2	I_3	I_4	I_5	I_6
U_7	2	1	4	0	1	0
U_8	1	2	1	0	2	5
U_9	4	1	2	1	1	0
U_{10}	1	4	1	0	4	0

Fig. 2.1 (**a**) Training set ($n \times m$), (**b**) test set

set [9] is dense; in contrast the Movielens data sets [10] are relatively sparse. The degree of sparsity, however, depends on the application type. To provide particular conclusions, someone has to experiment with the amount of sparsity as well.

Scalability: Scalability is important, since the number of users/items is very large in real-world applications. Related work [27] has proposed the use of dimensionality reduction techniques, which introduce a trade-off between the accuracy and the execution time of CF algorithms.

Train/Test Data Size: There is a clear dependence between the training set's size and the accuracy of CF algorithms [28]. Additionally, after an upper threshold of the training set size, the increase in accuracy is less steep. However, the effect of overfitting is less significant compared to general classification problems. In contrast, small training set sizes impact accuracy in a negative way. Therefore, a fair evaluation of CF algorithms should be based on adequately large training sets. Though most related research uses a size around 80 %, there exist works that use significantly smaller sizes [18]. From our experimental results we concluded that an 75 % training set size constitutes an adequate choice. But we have to notice that the training/test size should not be data set independent. (In the running example, we set training size at 60 %.)

Neighborhood Size: The number, k, of nearest neighbors used for the neighborhood formation results in a tradeoff: a very small k leads to low accuracy, because there are not enough neighbors to base the prediction. In contrast, a very large k impacts precision too, as the particularities of user's preferences can be blunted due to the large neighborhood size. In most related works [10, 26], k has been examined in the range of values between 10 and 100. The optimum k value depends on the data characteristics (e.g., sparsity). Therefore, for better tuning CF algorithms should be evaluated against varying k (in the running example, we have set $k = 3$).

Similarity Measure: Related works [11, 18, 19, 28] have mainly used Pearson correlation and cosine similarity. In particular, user-based (UB) CF algorithms use the Pearson correlation (Eq. (2.1)),[1] which measures the similarity between two users, u and v. Item-based (IB) CF algorithms use a variation of adjusted cosine-similarity (Eq. (2.2)),[2] which measures the similarity between two items, i and j, and has been proved to be more accurate [18, 28], as it normalizes the bias from subjective ratings.

$$\text{sim}(u,v) = \frac{\sum_{\forall i \in S} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{\forall i \in S} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{\forall i \in S} (r_{v,i} - \overline{r}_v)^2}}, \quad S = I_u \cap I_v. \tag{2.1}$$

$$\text{sim}(i,j) = \frac{\sum_{\forall u \in T} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{\forall u \in U_i} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{\forall u \in U_j} (r_{u,j} - \overline{r}_u)^2}}, \quad T = U_i \cap U_j. \tag{2.2}$$

[1] Means $\overline{r}_u$, $\overline{r}_v$ are the mean ratings of u and v over their co-rated items.

[2] Means $\overline{r}_u$, $\overline{r}_v$ are taken over all ratings of u and v.

a

	U_1	U_2	U_3	U_4	U_5	U_6
U_7	-0.19	0.19	1	-0.76	0.33	-0.14
U_8	-0.5	0.44	0.38	1	-0.82	0.67
U_9	0.41	-0.94	0.14	-0.47	1	-0.95
U_{10}	-0.5	0.5	-0.76	1	-0.82	0.67

b

	$1^{st}\,NN$	$2^{st}\,NN$	$3^{st}\,NN$
U_7	$U_3(1)$	$U_5(0.33)$	$U_2(0.19)$
U_8	$U_4(1)$	$U_6(0.67)$	$U_2(0.44)$
U_9	$U_5(1)$	$U_1(0.41)$	$U_3(0.14)$
U_{10}	$U_4(1)$	$U_6(0.67)$	$U_2(0.5)$

c

	I_1	I_2	I_3	I_4	I_5	I_6
I_1	-	-0.65	-0.66	0.36	-0.68	-0.42
I_2	-0.65	-	0.18	-0.51	0.50	-0.36
I_3	-0.66	0.18	-	-0.66	0.10	0.67
I_4	0.36	-0.51	-0.66	-	0	0
I_5	-0.68	0.50	0.10	0	-	0
I_6	-0.42	-0.36	0.67	0	0	-

d

	$1^{st}\,NN$	$2^{st}\,NN$	$3^{st}\,NN$
I_1	$I_4(0.36)$	-	-
I_2	$I_5(0.50)$	$I_3(0.18)$	-
I_3	$I_6(0.67)$	$I_2(0.18)$	$I_5(0.10)$
I_4	$I_1(0.36)$	-	-
I_5	$I_2(0.50)$	I3(0.10)	-
I_6	$I_3(0.67)$	-	-

Fig. 2.2 (**a**) Users' similarities matrix, (**b**) users' nearest neighbors in descending similarity matrix, (**c**) items' similarities matrix, (**d**) items' nearest neighbors in descending similarity matrix

Herlocker et al. [11] proposed a variation of the previous measures, which henceforth is denoted as Weighted Similarity (WS). If sim is a similarity measure (e.g., Pearson or cosine), then WS is equal to $\frac{\max(c,\gamma)}{\gamma} \cdot sim$, where c is the number of co-rated items.

Equation (2.1) takes into account only the set of items, S, that are *co-rated* by both users. This, however, ignores the items rated by only one of the two users. The number of the latter items denotes how much their preferences differ. Especially for the case of sparse data, by ignoring these items we discard significant information. Analogous reasoning applies for Eq. (2.2), which considers (in the numerator) only the set of users, T, that both co-rated the examined pair of items. The same applies for WS, which is based on Eqs. (2.1) or (2.2). To address the problem, in the following, we will examine alternative definitions for S and T.

The application of Pearson Correlation (Eq. (2.1)) to the running example is depicted in Fig. 2.2a and the resulting k-nearest neighbors(k-NN) are given in Fig. 2.2b. Respectively, the similarities between items, calculated with the adjusted cosine measure (Eq. (2.2)), are given in Fig. 2.2c, whereas Fig. 2.2d depicts the nearest neighbors. As only positive values of similarities are considered during the neighborhood formation, the items have different neighborhood size.

2.2.2 Second Stage Factors

Recommendation List's Size: The size, N, of the recommendation list results in a tradeoff: with increasing N, the absolute number of relevant items (i.e., recall) is expected to increase, but their ratio to the total size of the recommendation list (i.e., precision) is expected to decrease. (Recall and precision metrics are detailed in the following.) In related work [15, 28], N usually takes values between 10 and 50. (In the running example, we set $N = 2$.)

a

```
Vector GenTopMF(u, NN(u), I, N)
//User u
//Set NN(u)
//int I,N
begin
    for i = 1 to I
        f[i] = 0;
    foreach ν ∈ NN(u)
        for i = 1 to I
            if r[ν][i] ≥ Pτ
                f[i] = f[i] + 1;
    sort(f); //descending order
    n = 0; i = 1;
    while (n < N and f[i] > 0)
        A[n] = i;
        n = n + 1;
    return A;
end.
```

b

```
Vector GenTopMF(u, NN(i), I, N)
//User u
//Set NN(i)
//int I,N
begin
    for j = 1 to I
        f[j] = 0;
    for j = 1 to I
        if r[u][j] ≥ Pτ
            foreach ν ∈ NN(i)
                f[ν] = f[ν] + 1;
    sort(f); //descending order
    n = 0; i = 1;
    while (n < N and f[i] > 0)
        A[n] = i;
        n = n + 1;
    return A;
end.
```

Fig. 2.3 Generation of top-N list based on most frequent algorithm for (**a**) user-based algorithm and (**b**) item-based algorithm

Positive Rating Threshold: It is evident that recommendations should be "positive". It is not useful to recommend an item that will be rated with 1 in scale 1–5. Nevertheless, this issue is not clearly defined in several works. We argue that "negatively" rated items should not contribute to increasing the accuracy, and we use a rating-threshold, P_τ, to recommended items with rating no less than this value. If not a P_τ value is used, then the results may become misleading, since negative ratings can contribute to the measurement of accuracy.

Recommendation List Creation: The most often used technique for the generation of the top-N list counts the frequency of each item inside the found neighborhood, and recommends the N most frequent ones [26]. Henceforth, this technique is denoted as Most-Frequent item recommendation (MF). MF can be applied to both user-based and item-based CF algorithms. For example, assume that we follow the aforementioned approach for the test user U_7, for $k = 3$ and $N = 2$. For the case of user-based recommendation, the top-2 list includes items I_3, I_6. In contrast, for the case of item-based recommendation, the top-2 list includes 2 items of I_6 or I_2 or I_5 because all three have equal presence. Figure 2.3 describes the corresponding two algorithms (for the user and item-based CF, respectively).

We have to mention that these two algorithms, in contrast to the past work, in addition include the concept of positive rating threshold ($P\tau$). Thus, "negatively" rated items do not participate in the top-N list formation. Moreover, it is obvious that the generation of top-N list for the user-based approach is more complex and time consuming. The reason is that the former algorithm finds, firstly, user neighbors in the neighborhood matrix and then counts presences of items in the user-item matrix. In contrast, with the item-based approach the whole work is completed in the item neighborhood matrix.

Karypis [15] reports another technique, which additionally considers the degree of similarity between items. This takes into account that the similarities of the k neighbors may vary significantly. Thus, for each item in the neighborhood, this technique counts not just their number of appearances, but the similarity of neighbors as well. The N items with the highest sum of similarities are finally recommended. Henceforth, this technique is denoted as Highest-Sum-of-Similarities item recommendation (HSS). HSS is applicable only to item-based CF. For our example, the top-2 list based on this algorithm includes the items I_6, I_2, because they have the greater sum of similarities. Note that for both techniques, we have to remove from the recommendation list, these items that are already rated from the test user.

2.2.3 Third Stage Factors

Evaluation Metrics: Several metrics have been used for the evaluation of CF algorithms in related works [11, 12]: for instance the Mean Absolute Error (MAE) or the Receiving Operating Characteristic (ROC) curve. Although MAE has been used in most of these works, it has received criticism as well [18]. MAE is able to characterize the accuracy of prediction, but is not indicative of the accuracy of recommendation, as algorithms with worse MAE many times produce more accurate recommendations than others with better MAE. Since in real-world recommender systems the experience of users mainly depends on the accuracy of recommendation, MAE may not be the preferred measure. Other extensively used metrics are *precision* and *recall*. These metrics are simple, well known, and effective to measure the accuracy of recommendation procedure.

For a test user that receives a top-N recommendation list, let R denote the number of *relevant recommended items* (the items of the top-N list that are rated higher than P_τ by the test user). We define the following:

- *Precision* is the ratio of R to N.
- *Recall* is the ratio of R to the total number of relevant items for the test user (all items rated higher than P_τ by her).

Notice that with the previous definitions, when an item of the top-N list is not rated at all by the test user, it is considered as *irrelevant* and counts negatively to precision (as we divide by N). In the following, we also use $\mathrm{F}_1 = 2 \cdot \mathrm{recall} \cdot \mathrm{precision} / (\mathrm{recall} + \mathrm{precision})$. F_1 is used because it combines both the previous metrics.

Past/Future Data: In real-world applications, recommendations are derived only from the currently available ratings of the test user. However, in most of related works, all the ratings of each test user is considered a priori known. For a more realistic evaluation, recommendations should consider the division of the items of the test user into two sets [14]: (1) the past items of the test user, and (2) the future items of the test user.

2.3 Brief Literature Review

In 1992, the Tapestry system [8] introduced Collaborative Filtering (CF). In 1994, the GroupLens system [24] implemented a CF algorithm based on common users preferences, known as user-based CF algorithm, because it employs users' similarities for the formation of the neighborhood of nearest users. Since then, many improvements of user-based algorithm have been suggested, e.g., [3, 10].

In 2001, another CF algorithm was proposed. It was based on the items' similarities for a neighborhood generation of nearest items [15, 28] and is denoted as item-based CF algorithm.

Most related work followed the two aforementioned directions (i.e., user-based and item-based). Herlocker et al. [11] weight similarities by the number of common ratings between users/items. Deshpande and Karypis [6] apply item-based CF algorithm combined with conditional-based probability similarity and Cosine Similarity. Xue et al. [33] suggest a hybrid integration of aforementioned algorithms (memory-based) with model-based algorithms.

All aforementioned algorithms are memory-based. Their efficiency is affected from scalability of data. This means that they face performance problems, when the volume of data is extremely huge. To deal with this problem, many model-based algorithms have been developed [3]. However, there are two conflicting challenges. If an algorithm spends less execution time, this should not influence its quality. The best outcome would be to improve quality with the minimum calculation effort.

Furnas et al. [7] proposed Latent Semantic Indexing (LSI) in the area of Information Retrieval to deal with the aforementioned challenges. More specifically, LSI uses SVD to capture latent associations between the terms and the documents. SVD is a well-known factorization technique that factors a matrix into three matrices. Berry et al. [2] carried out a survey of the computational requirements for managing (e.g., folding-in[3]) LSI-encoded databases. They claimed that the reduced-dimensions model is less noisy than the original data.

Sarwar et al. [27, 29] applied dimensionality reduction for the user based CF approach. They also used SVD for generating predictions. In contrast to our work, Sarwar et al. [27, 29] do not consider two significant issues: (1) predictions should be based on the users' neighbors and not on the test (target) user, as the ratings of the latter are not a priori known. For this reason we rely only on the neighborhood of the test user. (2) The test users should not be included in the calculation of the model, because they are not known during the factorization phase. For this reason, we introduce the notion of pseudo-user to include a new user in the model (folding in), from which recommendations are derived. Other related work also includes Goldberg et al. [9], who applied Principal Components Analysis (PCA) to facilitate off-line dimensionality reduction for clustering the users, and therefore, achieves

[3]Folding in terms or documents is a simple technique that uses existing SVD to represent new information.

rapid on-line computation of recommendations. Hofmann [13] proposed a model-based algorithm, which relies on latent Semantic and statistical models. Moreover, Symeonidis et al. [30, 31] proposed a novel CF algorithm, which uses Latent Semantic Indexing (LSI) to detect rating trends and performs recommendations accordingly. Recently, Koren [16, 17], who is member of the winning team in the Netflix prize, proposed SVD++ method, which adds in the plain SVD also information taken from user/item bias and other implicit feedback. As the Netflix prize competition has demonstrated, matrix factorization models are superior to classic nearest-neighbor techniques.

2.4 Recommender Systems Paradigms

This section presents and briefly analyzes three Web Sites based on the three parallel approaches that have emerged in the context of recommender systems (i.e. CF, CB and hybrid methods).

CF exploits other users and their preferences to justify a recommendation to the target user. Usually the explanation is of the type "Customers who bought/rated item X also bought/rated items $Y, Z, \ldots$". It relies on the premise that both the target user and the users, which were used as an explanation have similar interests. A representative commercial recommender system that provides such recommendations is the online e-Commerce store Amazon.com.[4] As shown in Fig. 2.4, the user is presented with similar items that other customers have chosen to buy. The system assumes that the user is viewing an item, which they are already interested in. The Amazon system then finds similar users, who have already bought that specific item, and recommends the items that they bought to the user.

Fig. 2.4 Recommendation in Amazon

[4] http://www.amazon.com

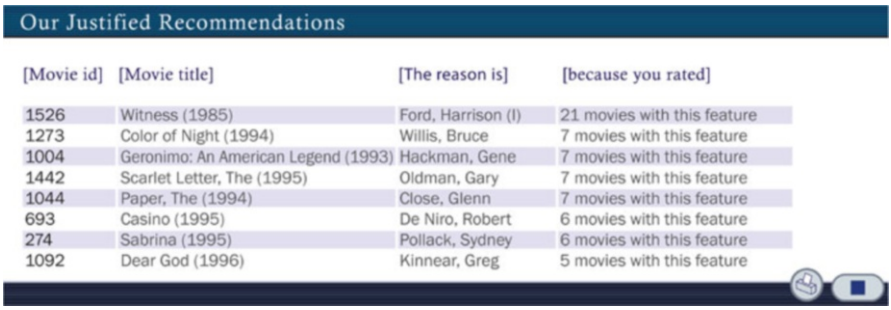

Fig. 2.5 Recommendations in MoviExplain

One example of the combination of content with rating data is MovieExplain[5] [32], which combines the rating user profile and the feature item profile to reveal the favorite features of users. MoviExplain builds a feature profile for each user and provides as explanation, the feature that influenced most a recommendation, showing also how strong is this feature in the feature profile of a user. As shown in Fig. 2.5, the link "The reason is" reveals the favorite feature that influenced most the MoviExplain's recommendations, whereas the link "because you rated" shows how strong is this feature in the feature profile of a user.

In recent years, new innovations in online Social Networks have encouraged more sharing between users even of different networks. The recommendations are based on the common network that two users belong to. The most striking of these innovations is Facebook Login (formerly Facebook Connect). The way it works is that partner firms install Login buttons and plugins on their websites and devices, which give Facebook users automatic access to information about their friends' activities. Such an example is the HuffPost Social News. HuffPost[6] is a site run by the Huffington Post, a well known American blog, where Facebook users can see what their friends have been reading and exchange stories and comments about them. The system's user interface is shown in Fig. 2.6.

The personalized HuffPost Social News pages create a forum for users to converse about news stories they have read, and in some cases add their relevant information for Facebook friends to read. The Huffington Post creates a social news experience with the "Recommendations" plugin on its home page, showing users personalized recommendations along with explanations. The system's user interface is shown in Fig. 2.6. The post recommendations (i.e. blog stories) are explained based on both the preferences of a user's friends and the ratings that these items have received.

[5] http://delab.csd.auth.gr/MoviExplain

[6] http://www.huffingtonpost.com/

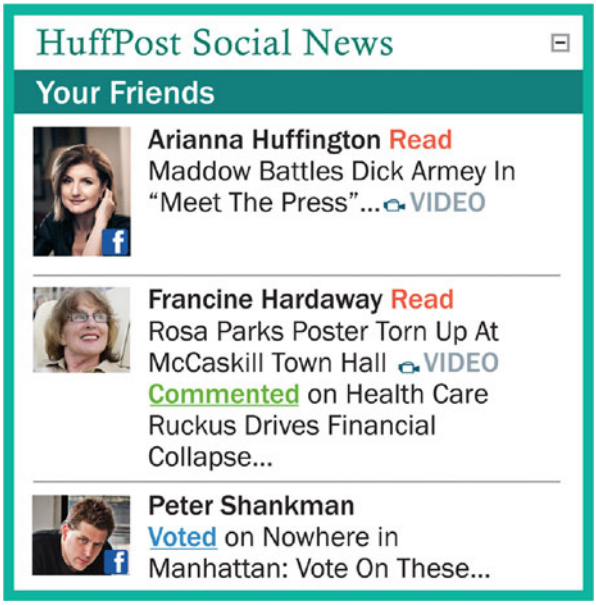

Fig. 2.6 The HuffPo Social News interface combined with Facebook

2.5 Social and Economic Report

The basic premise of recommenders is to reduce noise and filter out information, which is not relevant to the user taste. From the social point of view, the use of recommenders help people gain access to products or services that match their tastes. However, recommenders enforce users to insert information in log files of third party servers, arising privacy issues.

Regarding economy, there are several paradigms such as the Amazon.com, the Netflix.com and the Google.com recommendation engines, which have been proved as highly profitable. In particular, Amazon.com [5] claims that 35 % of products sales result from recommendations. Moreover, almost 66 % of movies rented in Netflix.com are recommended and Google News Recommendations generate 38 % more click-throughs [5].

In the same direction, friend recommendations in social networks stimulates users to expand their social circle, which is absolutely critical to retaining them and increasing the power of the network itself. Eventually, this brings income to a social network, based on the fact that companies invest their marketing budget except from target markets as well in mass power social networks. Recently, the incorporation of location to the recommendation process allows businesses to investigate new ways of profit. For instance, Foursquare.com incorporated a venue recommender in its mobile app. This recommender aims at offering to businesses a great advertising channel using their location, distance and users check-in history logs to stimulate users to visit the place.

References

1. M. Balabanovic, Y. Shoham, Fab: content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997)
2. M. Berry, S. Dumais, G. O'Brien, Using linear algebra for intelligent information retrieval. SIAM Rev. **37**(4), 573–595 (1994)
3. J. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, Madison, WI (1998), pp. 43–52
4. R. Burke, Hybrid recommender systems: survey and experiments. User Model. User-adapt. Interact. **12**(4), 331–370 (2002)
5. O. Celma, P. Lamere, Music recommendation tutorial, in *International Conference on Music Information Retrieval (ISMIR 2007)*, Vienna (2007)
6. M. Deshpande, G. Karypis, Item-based top-*n* recommendation algorithms. ACM Trans. Inf. Syst. **22**(1), 143–177 (2004)
7. G. Furnas, S. Deerwester, S. Dumais, Information retrieval using a singular value decomposition model of latent semantic structure, in *Proceedings of the 13th ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR)*, Grenoble (1988), pp. 465–480
8. D. Goldberg, D. Nichols, M. Brian, D. Terry, Using collaborative filtering to weave an information tapestry. Commun. ACM **35**(12), 61–70 (1992)
9. K. Goldberg, T. Roeder, T. Gupta, C. Perkins, Eigentaste: a constant time collaborative filtering algorithm. Inf. Retr. **4**(2), 133–151 (2001)
10. J. Herlocker, J. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in *Proceedings of the 22th ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR)*, Berkeley, CA (1999), pp. 230–237
11. J. Herlocker, J. Konstan, J. Riedl, An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Inf. Retr. **5**(4), 287–310 (2002)
12. J. Herlocker, J. Konstan, L. Terveen, J. Riedl, Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1), 5–53 (2004)
13. T. Hofmann, Latent semantic models for collaborative filtering. ACM Trans. Inf. Syst. **22**(1), 89–115 (2004)
14. Z. Huang, H. Chen, D. Zeng, Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. ACM Trans. Inf. Syst. **22**(1), 116–142 (2004)
15. G. Karypis, Evaluation of item-based top-n recommendation algorithms, in *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM)* (2001), pp. 247–254
16. Y. Koren, Collaborative filtering with temporal dynamics, in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Paris (2009), pp. 447–456
17. Y. Koren, Collaborative filtering with temporal dynamics. Commun. ACM **53**(4), 89–97 (2010)
18. R. McLauglin, J. Herlocher, A collaborative filtering algorithm and evaluation metric that accurately model the user experience, in *Proceedings of the 27th ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR)*, Sheffield (2004), pp. 329–336
19. B. Mobasher, H. Dai, T. Luo, M. Nakagawa, Improving the effectiveness of collaborative filtering on anonymous web usage data, in *Proceedings of the IJCAI Workshop on Intelligent Techniques for Web Personalization (ITWP)*, Seattle, WA (2001), pp. 53–60
20. R. Mooney, L. Roy, Content-based book recommending using learning for text categorization, in *Proceedings of the 5th ACM Conference on Digital Libraries (DL)*, San Antonio, TX (2000), pp. 195–204

21. M. O'Mahony, N. Hurley, N. Kushmerick, G. Silvestre, Collaborative recommendation: a robustness analysis. ACM Trans. Internet Technol. **4**(4), 344–377 (2004)
22. A. Papadimitriou, P. Symeonidis, Y. Manolopoulos, A generalized taxonomy of explanation styles for traditional and social recommender systems. Data Min. Knowl. Discov. **24**(3), 555–583 (2012)
23. M. Pazzani, D. Billsus, Adaptive web site agents. Auton. Agent Multi Agent Syst. **5**(2), 205–218 (2002)
24. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, Grouplens: an open architecture for collaborative filtering on netnews, in *Proceedings of the ACM Conference Computer Supported Collaborative Work (CSCW)*, Chapel Hill, NC (1994), pp. 175–186
25. J. Salter, N. Antonopoulos, Cinemascreen recommender agent: combining collaborative and content-based filtering. Intell. Syst. Mag. **21**(1), 35–41 (2006)
26. B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Analysis of recommendation algorithms for e-commerce, in *Proceedings of the ACM Conference on Electronic Commerce (EC)*, Minneapolis, MN (2000), pp. 158–167
27. B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Application of dimensionality reduction in recommender system - a case study, in *Proceedings of the ACM SIGKDD Workshop on Web Mining for E-Commerce - Challenges and Opportunities (WEBKDD)*, Boston, MA (2000)
28. B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in *Proceedings of the 10th International Conference on World Wide Web (WWW)*, Atlanta, GA (2001), pp. 285–295
29. B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Incremental singular value decomposition algorithms for highly scalable recommender systems, in *Proceedings 5th International Conference on Computer and Information Technology (ICCIT)*, Dhaka (2002), pp. 27–28
30. P. Symeonidis, A. Nanopoulos, A. Papadopoulos, Y. Manolopoulos, Scalable collaborative filtering based on latent semantic indexing, in *Proceedings of the 21st AAAI Workshop on Intelligent Techniques for Web Personalization (ITWP)*, Boston, MA (2006), pp. 1–9
31. P. Symeonidis, A. Nanopoulos, A. Papadopoulos, Y. Manolopoulos, Collaborative recommender systems: combining effectiveness and efficiency. Expert Syst. Appl. **34**(4), 2995–3013 (2008)
32. P. Symeonidis, A. Nanopoulos, Y. Manolopoulos, Moviexplain: a recommender system with explanations, in *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys)*, New York, NY (2009), pp. 317–320
33. G. Xue, C. Lin, Q. Yang, W.S. Xi, H.J. Zeng, Y. Yu, Z. Chen, Scalable collaborative filtering using cluster-based smoothing, in *Proceedings of the 28th ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR)*, Salvador (2005), pp. 114–121

http://www.springer.com/978-1-4939-0285-9

Recommender Systems for Location-based Social Networks

Symeonidis, P.; Ntempos, D.; Manolopoulos, Y.

2014, V, 108 p. 41 illus., 33 illus. in color., Softcover
ISBN: 978-1-4939-0285-9