Hydra: A Hybrid Recommender System

[Cross-Linked Rating and Content Information]

Stephan Spiegel
DAI-Labor
Technische Universität Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
spiegels@cs.tu-berlin.de

Jérôme Kunegis
DAI-Labor
Technische Universität Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
kunegis@dai-lab.de

Fang Li
CS & EE Department
Shanghai Jiaotong University
Dongchuan Road 800
200240 Shanghai, China
fli@sjtu.edu.cn

ABSTRACT

This paper discusses the combination of *collaborative* and *content-based* filtering in the context of web-based recommender systems. In particular, we link the well-known *MovieLens* rating data with supplementary *IMDB* content information. The resulting network of user-item relations and associated content features is converted into a unified mathematical model, which is applicable to our underlying neighbor-based prediction algorithm. By means of various experiments, we demonstrate the influence of supplementary user as well as item features on the prediction accuracy of *Hydra*, our proposed hybrid recommender. In order to decrease system runtime and to reveal latent user and item relations, we factorize our hybrid model via singular value decomposition (*SVD*).

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Information networks*

General Terms

Algorithm, Performance, Reliability

Keywords

Hybrid Recommender, Cross-Linked Data, Matrix Factorization

1. INTRODUCTION

Due to the enormous amount of information available online, the need for highly developed personalization and filtering systems is growing permanently. Recommender systems constitute a specific type of information filtering that attempt to present items according the interests expressed by a user [1]. Most web recommenders are employed for e-commerce applications or customer adapted websites, which assist users in decision making by providing personalized information [5].

In general, there exist two basic types of recommendation techniques, namely *content-based filtering* and *collaborative filtering*. Whereas content-based filtering methods examine items previously favored by the actual user [7], collaborative filtering techniques compute recommendations based on the information about similar items or users [10]. In our work we combine both strategies into one hybrid approach [3], which utilizes supplementary content features in order to improve the prediction accuracy of traditional collaborative filtering [6, 11].
For the development and evaluation of our proposed hybrid recommender system (also referred to as *Hydra*), we make use of the well-known *MovieLens*[1] rating data as well as the *IMDB*[2] movie database. Both corpora are joined in a unified mathematical model, which describes the complex network of interdependencies. Our model is easy to extend and enables us to extract latent user/item relations by means of matrix factorization.
The rest of the paper is structured as follows. Preliminary Section 2 discusses the combination of rating and content data. In Section 3, we describe the mathematical approach of our hybrid recommender. Prediction accuracy and system runtime are evaluated in Section 4. Finally, we conclude in Section 5.

2. DATA MODEL

For our research we investigate the *MovieLens* dataset, which contains 100,000 ratings for 1682 movies given by 943 users. The *GroupLens Research Group*[3] provides a disjoint training and test set (90% to 10% split) of the rating data, which enables us to evaluate the accuracy of our developed prediction algorithm. Ratings are initially presented as a list of triples (<userID,itemID,rating>), but can be converted into a sparse matrix format easily. Even though all users are anonymized, the *MovieLens* dataset contains demographic information that describe the individuals (e.g. age, gender and occupation) [11]. These content features are utilized by our hybrid recommender in the prediction step.
Beside user features, we furthermore employ supplementary item features that characterize the rated movies (e.g. date, country and genre). In our own approach item features are retrieved from *IMDB*, which represents a huge collection of movie related information. The *Internet Movie Database* is accessible online or can be downloaded as plain text files. In order to avoid performances loss caused by potentially unreliable network connections, we decided to hold a copy of the data files on our local system. Each of the numerous

[1]MovieLens Recommendations - http://movielens.umn.edu
[2]Internet Movie Database - http://www.imdb.com
[3]GroupLens Research - http://www.grouplens.org

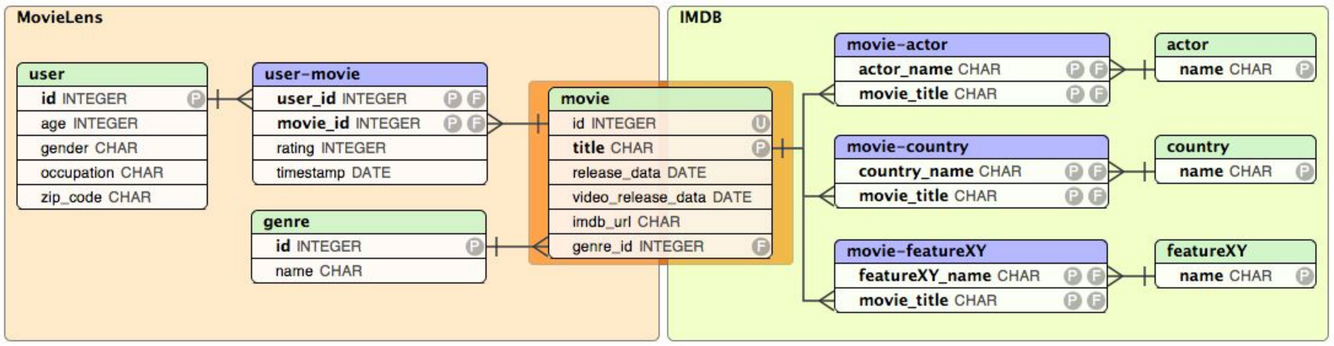


Figure 1: ER-Diagram of MovieLens and IMDB Data

IMDB files contains information about an independent item feature, whereas **not** every feature is practical to our hybrid approach.
In the following we are going to discuss the linkage of both *MovieLens* rating and *IMDB* content information.

2.1 Entity Relation

Figure 1 illustrates how the data entities of the *MovieLens* and *IMDB* corpus are interconnected. Obviously the movie entity acts as a central interface between both examined data sets, whereas all other data entities are arranged around the interface in star-shape. Note that only those entities highlighted in green color contain relevant user or rather item content features. Blue colored objects just represent connections between the actual data entities. The *user-movie* connection in Figure 1 can be considered as a special case, because every single link is characterized by an extra rating weight and time-stamp.

2.2 Bipartite Graph

Due to the fact that there does **not** exist any odd-length cycles between data entities (vertices are 2-colorable), our *ER*-Diagram can also be represented as a bipartite graph.

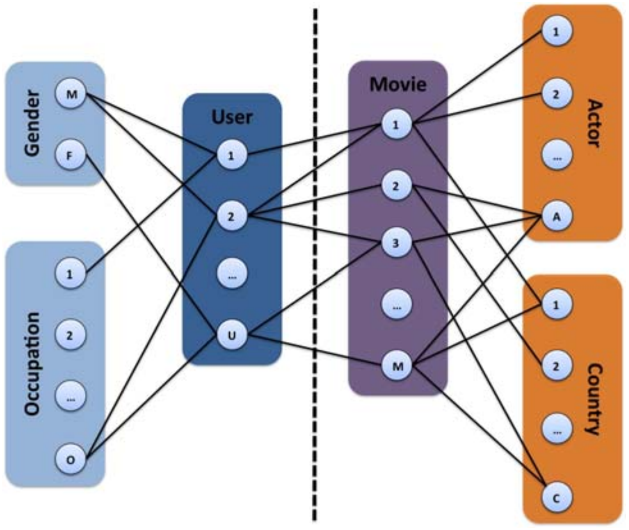
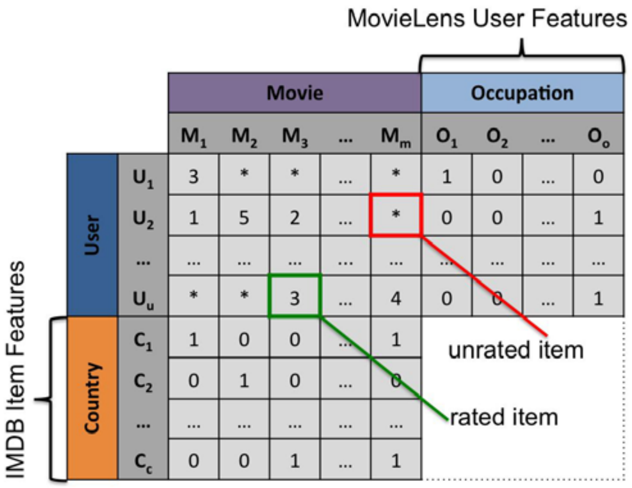


Figure 2: Bipartite Graph of Feature Relations

A possible presentation of the graph is shown in Figure 2, where only a small part of the entire network is illustrated. The single elements within our entities are also referred to as nodes, and are interconnected through so-called edges. In case of our bipartite graph, edges always link nodes of **unlike** colored entities (e.g. user $\leftrightarrow$ movie). This network structure allows us a straightforward combination of rating data and user/item features (refer to Section 3.2). In order to make the network information usable for data mining techniques, we need to transform the existing entity relations into an appropriate mathematical model.

2.3 Data Transformation

Figure 3 illustrates the *User-Movie* ratings as well as the *Movie-Country* and *User-Occupation* feature relations of our original bipartite graph in adjacency matrix notation. This unified model of rating and content information constitutes the core of our hybrid recommender. The supplementary user and item features contribute to a more precise description of the plain *MovieLens* rating data and are expected to improve the prediction accuracy for unrated items (refer to Section 4.1). In fact, injecting additional features of one source (such as content information) into an algorithm designed to process data with a different source (such as traditional user-oriented or item-oriented collaborative recommendation) is referred to as *Feature Combination* [3, 6] in most literature.


MovieLens User Features
IMDB Item Features
unrated item
rated item

		Movie					Occupation			
		M_1	M_2	M_3	...	M_m	O_1	O_2	...	O_o
User	U_1	3	*	*	...	*	1	0	...	0
	U_2	1	5	2	...	*	0	0	...	1
	...	...	...	...	...	...	...	...	...	...
	U_u	*	*	3	...	4	0	0	...	1
Country	C_1	1	0	0	...	1				
	C_2	0	1	0	...	0				
	...	...	...	...	...	...				
	C_c	0	0	1	...	1				

Figure 3: Extended Rating Matrix

3. HYBRID RECOMMENDER

In this section we want to discuss rating prediction in terms of *Hydra*, our proposed hybrid recommender system. But first of all we want to give attention to data normalization, feature combination and matrix factorization, which are all preliminary steps to rating estimation.

3.1 Data Normalization

When we compare the items of our *MovieLens* rating matrix with entries of our generated feature matrices (see Figure 3), we directly notice that both exhibit a unlike range of values. Whereas movie ratings can range from 1 to 5 (zero if non-rated), content features are either existent or not (1 or 0). Consequently, original rating matrix ($R \in \{0,\ldots,5\}^{m\times n}$) and user/item feature matrices ($U \in \{0,1\}^{u\times y}; I \in \{0,1\}^{x\times i}$) need to be normalized differently.

3.1.1 *Subtractive Normalization*

Generally *user-item* ratings exhibit different kinds of global effects [2, 4]. For instance, some users always tend to give higher ratings on items than other users, as well as some items at an average receive more positive user feedback than other items. In order to compute accurate rating predictions these global effect need to be removed form our data [12].
Usually a weighted combination of overall-, user- and item-average rating values ($\overline{r}, \overline{r}_u$ & $\overline{r}_i$) is subtracted from the original entries (r_{ui}) to remove individual user preferences as well as item popularity effects [10, 6, 8]:

$$\tilde{r}_{ui} = r_{ui} - \alpha\overline{r} - \beta\overline{r}_u - \gamma\overline{r}_i \tag{1}$$

The parameters α, β and γ determine the influence of the observed effects on the final normalization result. We make use of the least square estimation to automatically learn all normalization parameters [4, 12].

3.1.2 *Multiplicative Normalization*

Since the entries of our generated feature matrices are either 0 or 1, all average values equal 1 ($\overline{r}, \overline{r}_u, \overline{r}_i = 1$). In case of subtractive normalization, the feature values would just be shifted instead of being regularized. Therefore we make use of multiplicative normalization, which regularizes all entries within a feature matrix F according their respective row and column length:

$$\tilde{F} = M \cdot F \cdot N \tag{2}$$

where the diagonals of the matrices M and N contain the specific row and column multipliers:

$$M_{xx} = \frac{1}{\sqrt{\sum_n F_{xn}}} \quad \text{and} \quad N_{yy} = \frac{1}{\sqrt{\sum_m F_{my}}} \tag{3}$$

Note that all off-diagonal entries within the multiplication matrices are zero, and do not affect the normalization.

3.2 Feature Combination

As illustrated in Figure 3, our original rating matrix R and our retrieved user and item feature matrices (U and I) are combined in a unified model:

$$H_{m\times n} = \begin{bmatrix} R_{u\times i} & U_{u\times y}\cdot w_2 \\ I_{x\times i}\cdot w_1 & 0_{x\times y} \end{bmatrix} \text{ with } \begin{pmatrix} m = u+x \\ n = i+y \end{pmatrix} \tag{4}$$

Note that we need to fill our extended matrix with zeros to preserve the original rectangular shape. Moreover we have to keep in mind that the dimensions of the supplementary feature matrices have to fit our original rating matrix (refer to Eq.(4)). Needless to say, the respective user or rather item entries need to be conform as well.
Equation (4) furthermore reveals that the retrieved user and item features are assigned additional weights (w_1 and w_2 respectively). By means of these weights we can directly control the influence of the individual features on the final rating estimate. Our purpose is to identify those features and appropriate weights, which can improve the prediction accuracy of our hybrid recommender system.
For our further discussion we assume that the hybrid model H merely holds normalized rating and feature information.

3.3 Matrix Factorization

Typically matrix factorization techniques are used to reduce the dimension of the item space and/or to retrieve latent relations between items of the observed dataset [9, 4]. In our work we want to investigate the sophisticated *Singular Value Decomposition (SVD)* approach, which factorizes our inflated rating matrix H into three low-dimensional matrices containing the left-singular vectors (V), the singular values (S) and right-singular vectors (W) respectively. In case of a reduction to dimension k, we can say that the product of the resulting matrices is a rank-k approximation of our extended rating matrix $H_{m\times n}$:

$$H_k = V_{m\times k} \cdot S_{k\times k} \cdot W^T_{k\times n} \tag{5}$$

The prediction accuracy of our designed hybrid recommender system strongly depends on the parameter k, which will be closely examined in Section 4.

3.4 Rating Prediction

Our generated matrices V, S and W can be utilized for the rating prediction step in several different ways [9]. Probably the most straightforward method to estimate an unknown user-item rating (r_{ui}) is, to simply multiply the according singular vectors and singular value (pure SVD approach):

$$\hat{r}_{ui} = [VS^{1/2}]_u \cdot [S^{1/2}W^T]_i \tag{6}$$

But in our actual approach (HYB-SVD-KNN) we consider the compound matrices $X = [VS^{1/2}]$ and $Y = [S^{1/2}W^T]$ as separate user and item concepts, which can be employed for user-oriented or rather item-oriented collaborative filtering. In case of item-oriented *CF*, missing user-item ratings are calculated based on known ratings given by the same user on similar items [10, 8]:

$$\hat{r}_{ui} = \frac{\sum_{j\subset ratedItems(u)} s_{ij}\cdot r_{uj}}{\sum_{j\in ratedItems(u)} |s_{ij}|} \tag{7}$$

where s_{ij} represents the *Cosine Similarity* of the according item vectors:

$$s_{ij} = \frac{Y_i \cdot Y_j}{||Y_i|| \cdot ||Y_j||} \tag{8}$$

In general, the prediction accuracy of a collaborative filtering system depends on the employed similarity measure as well as the number of examined neighborhood users/items that are involved in rating estimation [12]. Therefore, we are going to scrutinize the influence of the neighborhood size in the following section.

4. EVALUATION

As mentioned earlier, the prediction accuracy of our developed hybrid recommender depends on several different parameters, like neighborhood size, data dimension, utilized features and so on. This section will investigate the behavior of our system under certain conditions. As measurement for prediction accuracy we use the well-known *Root Mean Squared Error* [4], which computes how close the estimates (predictions based on the training-set) are to the values actually observed (test-set).
Finally we are going to compare the overall performance of *Hydra* (also referred to as HYB-SVD-KNN) with other implementation variants, at which prediction accuracy as well as computational effort are scrutinized.

4.1 Neighborhood Size

In general, there exist two basic implementation variants of the *KNN*-algorithm, namely user- and item-oriented collaborative filtering. Whereas user-oriented techniques just consider like-minded user to predict unknown ratings, item-oriented methods utilize similar items [10, 13]. However, both *CF* variants employ the same underlying mathematical approach, in which the current examined user/item rating is estimated based on the k-nearest neighbors. As illustrated in Figure 4, the prediction accuracy is strongly dependent on the number of neighbors taken into account.

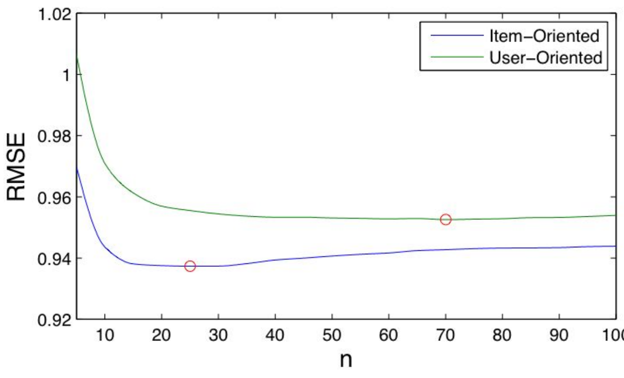


Figure 4: Pure KNN with n Neighbors

Obviously our item-oriented approach performs much better than the user-oriented implementation, which agrees with the general observations about *CF* systems that can be found in most literature [8, 2], and might be explained by the fact that individuals are more familiar with the items previously preferred than with potential like-minded users.
Figure 4 furthermore reveals that a relatively small neighborhood produces imprecise rating estimates, which is due to the fact that the considered users or rather items do not contain sufficient information to make reliable predictions. However, in case of a comparatively large neighborhood it might happen that too many users or items with very low similarities are taken into account [4, 6].
For our further analysis we will make use of the optimum neighborhood size determined for user-oriented and item-oriented *CF* respectively ($n = 25/70$).

[4]$RMSE(O,P) = \sqrt{\frac{sum_{i=1}^{n}(O_i-P_i)^2}{n}}$; where O and P are vectors of observed and predicted values respectively

4.2 Data Dimension

The prediction accuracy of our hybrid recommender system furthermore depends on the dimension k of the decomposed matrices (V, S and W; see Eq.(5)) [10, 4]. Whereas relatively short singular vectors do not have enough explanatory power to differentiate the appropriate users or items, comparatively long vectors might lead to over-fitting. These characteristics are confirmed by our analysis in Figure 5.

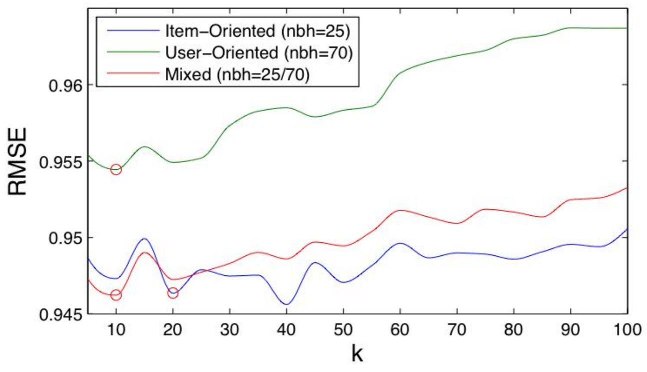


Figure 5: SVD-KNN in k-Dimensional Space

As before, the item-oriented implementation produces better prediction results than the user-oriented counterpart. The displayed mixed approach of our *SVD-KNN* algorithm can be described as a weighted combination[5] of user-oriented and item-oriented prediction results. However, this method could only achieve considerable performance improvements when supplementary user or rather item features were taken into account (see Figure 7).
According to the graphs in Figure 5, the global minimum of our user-oriented as well as mixed collaborative filtering approach is reached at a dimension of $k = 10$. For further studies on our item-oriented approach we set $k = 20$, because the dimension of the global minimum is quite high ($k = 40$) and produces an unsatisfactory system runtime.

4.3 Feature Weights

Besides analyzing the influence of neighborhood size and data dimension, we additionally investigate the effect of the retrieved user- and item features on the prediction accuracy of our designed hybrid recommender (HYB-SVD-KNN algorithm). As mentioned earlier, the contribution of the single user and item features is regulated by weights (refer to Eq.(4)). Figure 6 shows the behavior of *Hydra* under the influence of various weighted features, employing the optimal parameter settings determined previously ($n = 25/70$ and $k = 10/20$, (user-oriented/item-oriented CF)).
All user and item features are selected based on their matrix fill-rate, whereas only feature matrices with a low sparsity ($\leq 99\%$) are considered for rating prediction. This is due to the fact that sparse feature matrices do not increase the explanatory power of our original rating matrix R. In our research we investigate the *occupation*, *gender* and *age* of the examined *MovieLens* users as well as *country*, *date* and *genre* of the appropriate movies.
As we can see in Figure 6, the performance improvement[6] made by the examined movie features is much higher than by the considered user features. In particular the *Movie-Gender* feature produced

[5]70/30 ratio of user/item-oriented *CF* estimates
[6]$\Delta RMSE = RMSE(w_0) - RMSE(w_{opt})$

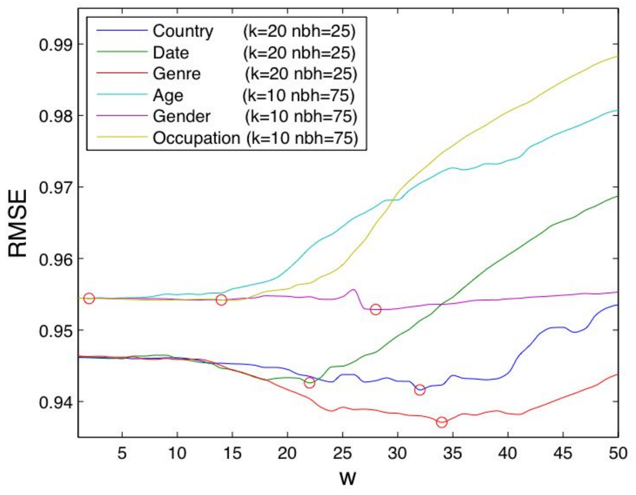


Figure 6: HYB-SVD-KNN with Weighted Features

surpassing prediction results. But the highest prediction accuracy could be achieved by the combination of both individual user and item features.
Figures 7 compares different single and combined feature variants of our novel *HYB-SVD-KNN* algorithm, in which the optimal determined feature weights are displayed in the respective bars. For both user-oriented and item-oriented *CF* approach, the combined feature variant performed best. However, the mixed implementation with merged user and item features, marked by the horizontal bottom-line, outperformed all other variants.

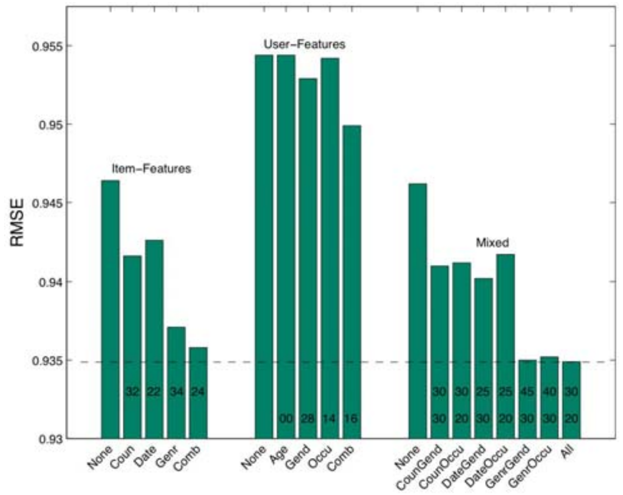


Figure 7: HYB-SVN-KNN with Combined Features

4.4 System Performance

The previous section demonstrated that our own *HYB-SVD-KNN* algorithm (also referred to as *Hydra* system) is able to raise prediction accuracy by incorporating weighted user and item features. In this section we want to compare *Hydra* with the pure *SVD* approach (refer to Eq.(6)) and the *KNN* strategy (traditional CF) as well as with the advanced *SVD-KNN* implementation (joined SVD and KNN approach). Besides contrasting the prediction accuracy of these algorithms, we furthermore want to scrutinize their runtime. Figure 8 shows a juxtaposition of the individual rating prediction methods, where the computational effort is presented on a logarithmic scale. Note that the algorithm runtime is directly dependent on the available processing power.
In general, the performance and runtime of recommender systems are in competition and have a mutual influence on each other. For this reason we need to treat a recommender with all its factors as a whole. According to Figure 8, our proposed hybrid recommender (*HYB-SVD-KNN* algorithm) is superior in terms of prediction accuracy. Although the pure *SVD* approach shows the lowest computational effort, our hybrid method is about four times faster than traditional collaborative filtering (*KNN* approach). On that account we believe that *Hydra* (alias *HYB-SVD-KNN* algorithm) is practical to real-life applications.

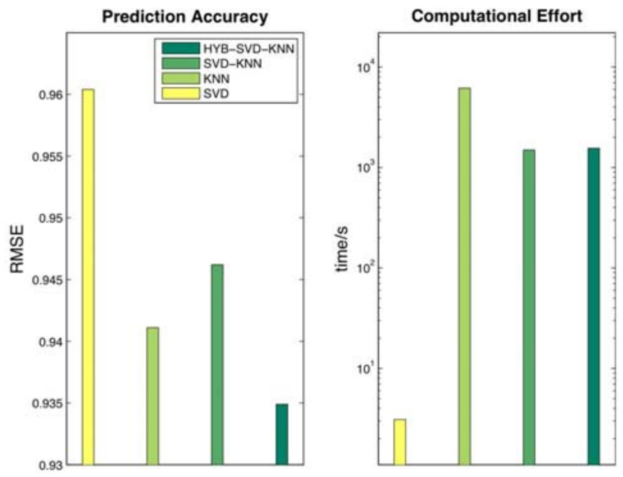


Figure 8: System Performance Overview

5. CONCLUSIONS

This paper proposes a novel approach to rating prediction, in which collaborative filtering and content-based filtering techniques are combined. Our hybrid approach is special in that rating data as well as content information are joined in a unified model, which leads to less parameters and more reasonable prediction results.
In particular, we describe the linkage of the well-known *MovieLens* rating data and the *IMDB* movie information. By means of various experiments, we demonstrate that the retrieved user and movie content features are beneficial to the prediction accuracy of traditional collaborative filtering algorithms.
For the purpose of minimizing the runtime of our designed hybrid recommender system as well as to extract latent user and movie relations, we factorize our unified model by means of singular value decomposition. The dimensionally reduced data can be employed to directly estimate unknown ratings (*pure SVD approach*) or rather to accelerate collaborative filtering (*SVD-KNN* as well as *HYB-SVD-KNN* algorithm).
All of the discussed implementation variants are compared in terms of prediction accuracy and computational effort, at which our proposed *HYB-SVD-KNN* algorithm exhibits the best overall performance. However, it would be interesting to study another dataset from a different domain, because unlike content features might achieve even higher prediction accuracy improvements.

6. REFERENCES

[1] Adomavicius and Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEETKDE: IEEE Transactions on Knowledge and Data Engineering*, 17, 2005.

[2] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 43–52, Washington, DC, USA, 2007. IEEE Computer Society.

[3] R. D. Burke. Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 377–408. Springer, 2007.

[4] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[5] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1), 2003.

[6] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *ACM SIGIR Workshop on Recommender Systems*, pages 187–192, 2002.

[7] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, chapter 10, pages 325–341. Springer-Verlag, Berlin, Germany, May 2007.

[8] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[9] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system - a case study. In *In ACM WebKDD Workshop*, 2000.

[10] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, chapter 9, pages 291–324. Springer-Verlag, Berlin, Germany, may 2007.

[11] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: Large scale online bayesian recommendations. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 111–120, New York, NY, USA, 2009. ACM.

[12] A. Toescher, M. Jahrer, and R. Legenstein. Improved neighborhood-based algorithms for large-scale recommender systems. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, Graz, Austria, 2008. ACM.

[13] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, New York, NY, USA, 2006. ACM Press.

APPENDIX

A. MATLAB IMPLEMENTATION

A.1 Factorization of Unified Model

```
\\load normalized rating and feature matrices
  load IO_File R U I;

\\load parameters
  load IO_File k;

\\assign features with weights
  U*w1; I*w2;

\\create fill matrix
  dim1 = numel(I(:,1)); dim2 = numel(U(1,:));
  Z = zeros(dim1,dim2);

\\join rating and feature information
  H = [R,U;I,Z];

\\factorize unified model
  [V,S,W] = svds(H,k);

\\create compound matrices (user/item-concept)
  X = V*sqrt(S); Y = sqrt(S)*W';

\\save results
  save -append IO_File X Y;
```

A.2 Computation of Item Similarities

```
\\load item-concept matrix and parameters
  load IO_File Y numItems;

\\create item similarity matrix
  ISM = zeros(numItems,numItems);

\\compute item sililarities
  for i = 1:numItems
    for j = 1:numItems
      if i <= j
        break;
      else
        ISM(i,j) = cosineSim(Y(:,i),Y(:,j));
      end
    end
  end
  ISM = ISM + ISM';

\\sort item similarities
  [S,IX] = sort(ISM,2,'descend');

\\save results
  save -append IO_File ISM IX;
```

A.3 Rating Prediction

```
\\load rating and item similarity matrix
  load IO_File R ISM IX;

\\load neighborhood size and user/item indices
  load IO_File n userU itemI;

\\create empty variables
  sum1 = 0; sum2 = 0;

\\predict unknown rating for userU and itemI
  for nb = 1:n
    itemJ = IX(itemI,nb);
    if (itemI ~= itemJ) & (R(userU,itemJ) ~= 0)
      itemSim = ISM(itemI,itemJ);
      sum1 = sum1 + (itemSim * R(userU,itemJ));
      sum2 = sum2 + abs(itemSim);
    end
  end
  pred(userU,itemI) = (sum1 / sum2);
```