Addressing the cold-start new-user Problem for Recommendation with Co-training

Jelena Slivka*, Aleksandar Kovačević*, Zora Konjović*
* University of Novi Sad/Faculty of Technical Sciences/Computing and Control Department, Novi Sad, Serbia
slivkaje@uns.ac.rs, kocha78@uns.ac.rs, ftn_zora@uns.ac.rs

***Abstract*—Many online domains rely on recommender systems for personalization of the content offered to their customers. In order to personalize the content for the user, recommender systems rely on user's rating history. When a new user is joining the system, the lack of rating history is a serious problem for building a personalized model. In this paper we strive to alleviate this new-user problem by reducing the number of user's ratings needed for quality recommendation. We pose the recommendation problem as a classification problem: for each user a separate classification model is trained. Given the item unrated by that user, this model predicts whether the user will like or dislike the item. In order to alleviate the new-user problem, we employ the semi-supervised co-training algorithm. The co-training algorithm assumes a multi-view setting (i.e. a defined split of features in the dataset). In this paper we propose to use features based on users' ratings as the first view and item description (content features) as the second view. We perform our experiments on a popular *MovieLens* dataset, and show that, by using co-training, for the users that have rated very few items; we can achieve the performance a supervised system would have, given a huge rating history for that user.**

I. INTRODUCTION

Users today are faced with an excessive number of choices, whether they are shopping, looking for restaurants, movies or even education. Thus, many online domains rely on recommender systems to personalize the desired content for each user [1]. *Amazon*[1] uses recommender algorithms to personalize shopping experience for customers [2]. *MovieLens*[2] gives personalized recommendations and rating prediction for the movies user has not rated [3]. *TripAdvisor*[3] is a travel search engine that assists consumers in search for travel information [4].

There are three major approaches for building recommender systems: content-based filtering, collaborative filtering, and hybrid filtering which combines the two former approaches. Collaborative filtering algorithms are based on users' similarity – the assumption is that the users with similar tastes will rate items similarly. On the other hand, Content-based filtering approaches offer recommendations based on the rating history of a user and item content relevance. The assumption of both approaches is based on user's rating history, and the lack of needed ratings when the new user joins the system can seriously hurt the performance of the recommender system. This is known as the *new-user cold-start* problem [5].

In this paper we strive to alleviate the new-user problem by reducing the number of user's ratings needed for accurate recommendation. In other words, for a given new user that has rated very few items, our goal is to achieve the performance the recommender system would have if it was provided a substantial amount of user's ratings.

We have posed the recommendation problem as a classification problem: for each given user-item pair we predict whether the user will "like" the item (i.e. we should recommend it) or "dislike" the item (i.e. we should not recommend it). For building the classification model we use the items previously rated by user as training data. For the new users we have a very few training examples, but, on the other hand, we also have a huge amount of unlabeled data (items that the user has not rated). This is an ideal setting for semi-supervised learning techniques that are, under certain conditions, capable of producing a high quality classifier from only a small amount of labeled data and a sufficiently large amount of unlabeled data. We have chosen to use a broadly used semi-supervised technique – the co-training [6] algorithm. Co-training implies a multi-view problem setting (i.e. the feature set of each example can be naturally partitioned into two distinct feature sets called views). This also suits our recommendation problem very well as we can easily define different views that describe the items. For example, information about users' ratings can be treated as the first view and item description can be treated as the second view.

In this paper we propose a novel multi-view, hybrid recommender system based on co-training that considers ratings given by other users' in the system as the first view (collaborative filtering predictor), and item description as a second view (content-based predictor).

In the initial experiments with our algorithm, we consider the movie recommendation problem, by using a subset of popular *MovieLens*[4] corpus. We use information about movie genre and movie plot description found on *IMDB*[5] as the content description for each movie.

In the paper we test several co training settings: standard co-training [6] run using a "natural" feature split, co-training run with random feature split, *Majority Vote* of several different co-training classifiers run with random feature split, and, finally, Random Split Statistic algorithm (RSSalg) we have developed earlier in [7] with

[1] http://www.amazon.com/
[2] http://www.movielens.org/
[3] www.tripadvisor.com
[4] http://files.grouplens.org/papers/ml-10m.zip
[5] http://www.imdb.com/

the goal of boosting the performance of co-training. We define several "natural" feature splits based on different view combinations. As the first view we propose to use the users' ratings, and, for the second view, we experiment with using just genre information, using just plot description, and using the combination of plot and genre features. We also do an additional experiment with a purely content-based recommender: we employ genre features as the first view and plot features as the second view. We show that, by employing co-training, for each feature split we are able to boost the performance of the initial classifier and even achieve the performance the supervised classifier would have given a huge rating history. The best performing settings in our experiments were both co-training and RSS algorithms when applied with users' ratings as the first view and the combination of genre and plot features as the second view, or users' ratings as the first view and the just genre features as the second view. We also show that using a purely content predictor constructed with plot and genre features can be very useful in combination with co-training if there are no available ratings from other users.

This paper is organized as follows. Section 2 presents the related work. Section 3 describes our methodology. Section 4 presents the experiments conducted in this paper and achieved results. Finally, section 5 concludes the paper and gives directions for future work.

II. RELATED WORK

As a first view in our multi-view setting we use a collaborative-filtering predictor. Authors in [8] treat CF as a classification problem and discretize ratings into a small number of classes. They define their CF method as a machine learning framework and build a separate model for each user in the database. We have adopted this approach for our CF predictor.

By applying semi-supervised learning techniques we are able to induce high quality classifiers from only a small amount of labeled examples, thus greatly reducing the manual work needed for labeling training sets. One major semi-supervised learning technique is co-training [6], which is based on multi-view learning. The goal of multi-view learning is to combine predictors derived from each of the views separately, in such a way that the resulting combined predictor outperforms predictors trained on each of the views separately, or predictors trained on trivial combinations of the views. Although both multi-view and semi-supervised learning seem to fit perfectly as a solution for cold-start problem in recommendation settings, there are very few papers that utilize them for recommendation problem [9].

In [9] authors develop a content-based movie recommendation system that integrates the content from three different data sources associated with movies (image, text and audio). Each data source, i.e. media type is considered to be a different view of the data in the designed multi-view framework. The authors employ co-training in order to enrich the user profile in case where there are only a few rating histories for a given user. Similar to [9], we also employ co-training with the same goal of alleviating the recommendation process for the new users; however there are several important differences between our work and work presented in [9]:

- Authors in [9] propose a purely content-based recommender system, while we propose a hybrid recommendation system that utilizes not only item description, but also rating histories of other users' in the system.
- In [9], single-view recommendations are based on finding user's *k* nearest neighbors and assigning the user's rating based on most frequent rating from the obtained nearest neighbor set. This rating is also assigned a prediction score based on the combination of distance measures between the given movie and its nearest neighbors and number of times the given rating was assigned to nearest neighbor movies. In this paper, we treat both item description and users' rating as features in the classification problem and employ a machine learning algorithm in order to predict the rating. Our proposed approach can be used with any classifier that can handle missing data.
- After single-view score assignment and recommendation in [9], multi-view profile enrichment is applied. In the co-training process, only the movies rated with the same score by all of the views are added to the training set. In our setting, standard co-training [6] is applied for user enrichment.

In [10] an idea to create a hybrid recommender system by combining content and social information with co-training in order to induce a more accurate model of human preferences is proposed. However, a concrete methodology or experimental results are not provided.

Authors in [11] develop a hybrid recommender system. They analyze product descriptions and user behaviors in order to automatically extract semantic attributes. The process of semantic attribute extraction is facilitated by semi-supervised learning. After extraction, a Naive Bayes classifier is used in order to implement a content-based recommender system. In contrast to [11], we employ semi-supervised learning directly in the context of recommender system training.

III. METHODOLOGY

In this section we describe the hybrid multi-view recommendation system proposed in this paper. As proposed in [10], the two views used in our framework to describe the data are:

- social information (i.e. users' ratings) which is used to construct a collaborative filtering predictor, and
- content information which is used to construct a content-based predictor.

In order to apply classic co-training [6] we have defined item recommendation as a classification problem. As in [8], we are trying to induce a model for each user separately, that will allow the classification of items unseen by that user in two classes – *Like* and *Dislike*. We discretize the rating value in these two classes by defining a rating value threshold *t* and treating the ratings that exceed this threshold as label *Like*, and the rest of the ratings as label *Dislike*.

In sections A and B we describe the way the two single-view predictors that will be used in our co-training setting are created. In section C we describe our co-training settings.

A. First view: A collaborative filtering predictor

Users' rating data can be represented as a sparse matrix where rows (training examples) correspond to items and columns (features) correspond to users' ratings for given items [8]. We refer to the matrix as sparse because most of the values are missing (users typically rate a small subset of all possible items). The value of attribute *u* for training example *i* corresponds to the rating[6] given by user *U* to item *I*. We will refer to the feature set constructed this way as *User view*.

The prediction task can be seen as filling in the missing values of the matrix. We build a separate model for each user by treating the corresponding user's rating feature as label. For each user, we use the items that the user has rated as training data for our model. The rest of the items (the user has rated) are used as examples for which we need to induce the label (user's rating for the item).

After the data is represented in the described way, we apply a machine learning algorithm that can tolerate missing values in order to train the model for each user[7]. However, for the new users that have rated just few of the items, the resulting model would be very weak due to a small number of training examples.

B. Second view: A content-based predictor

The second view of the data consists of features extracted from item description. In the experiments conducted in this paper, as the item description in the context of movies, we use text data (movie plot description, which we refer to as *Plot view*), list of movie genres (which we refer to as *Genre view*) and the combination of these two feature sets constructed by putting all of these features together (referred to as *Genre&Plot view*). Each genre that appears in the dataset is represented as a binominal feature that can have values *true* (the movie belongs to this genre) or *false* (the movie does not belong to this genre).

C. The applied co-training settings

In its original form, co-training is applicable to the dataset that has a natural partitioning of the features in two disjoint subsets (views) where each view is sufficient for learning and conditionally independent of the other view given the class label [6]. Co-training exploits the two views in order to train two classifiers using the available training examples. Then, iteratively, each classifier selects and labels some unlabeled examples in order to improve the accuracy of the other classifier by providing it with unknown information. This is an ideal setting for movie recommendation as we can easily obtain information about the movie from several different sources of information.

In this paper we experiment with several different "natural" feature splits:

- *User_Plot*: *User view* treated as the first view and *Plot view* treated as the second view.
- *User_Genre*: *User view* treated as the first view and *Genre view* treated as the second view.
- *User_Genre&Plot*: *User view* treated as the first view and *Genre&Plot view* treated as the second view.

We apply several different co-training settings:

- ***Natural***: standard co-training, as proposed in [6] applied with "natural" feature splits defined above,
- ***Random***: standard co-training applied with random feature split (obtained by randomly splitting all available features from both views in two feature sets)
- ***Majority Vote (MV)***: algorithm that constructs the ensemble of diverse co-training classifiers by creating a number of different random feature split and using them to train different co-training classifiers. *MV* combines the predictions of the obtained ensemble in a simple majority vote fashion [7].
- ***Random Split Statistic Algorithm (RSS)***: algorithm we have developed earlier in order to boost the performance of co-training and enable its application to single-view datasets [7]. In the same way as *MV*, *RSS* trains diverse co-training classifiers. The training set produced by each co-training process is different and it consists of initially labeled examples and examples labeled in co-training process. All of these co-training results are processed by selection of the examples that appear in most resulting training sets and for which most of the resulting co-training classifiers agree on the label. The final training set is formed from these selected results and it is used for learning a model with much higher classification performance then the initial model trained solely on labeled data. Finally, we will dente *RSS* optimized on the test data (upper bound performance for *RSS*) [7] as ***RSS_best***.

IV. Experimental results

For evaluating the performance of our solution, we have used a popular *MovieLens* dataset. We adopted a subset of this data for which the authors in [12] harvested content-descriptions from the IMDB Web site. We have processed the movie plot descriptions by applying tokenization, conversion to lower case, a stop-word filter and Porter's stemmer. From the words obtained this way we have built a dataset based on bag-of-words model, using the TF-IDF (term-frequency-inverse-document-frequency) measure as the value of the word in the obtained feature vector. We have discretized the ratings (1 to 5) in the following way: {1,2,3}→*Dislike*, {4,5}→*Like*.

In order to evaluate co-training performance we have used the stratified 10-fold-cross validation described in [7]. In the standard 10-fold-cross validation procedure the experimental data is divided in 10 folds, and in each of the 10 rounds, a different fold (10% of the data) is used for testing, while the remaining 9 folds (90% of the data) are used for training. Typically, co-training uses only a small amount of both labeled and unlabeled data, and applying the standard 10-fold-cross validation procedure on co-training results with many examples being omitted from both testing and training data. In order to better utilize the available data, the size of the test set is increased in order to improve the evaluation without significantly reducing the quality of the obtained classifier. Thus, we divide the data in 10 stratified folds. In each round of 10-fold-cross validation process, a different fold is selected for random selection of required

[6] Instead of actual rating value, we use our derived classes, i.e. *Like* and *Dislike*, because this approach yielded with a slightly better performance in our experiments

[7] Authors in [8] also propose a way to transform this representation in order to apply machine learning algorithms that cannot handle missing values

number of labeled training examples. The remaining data from that fold, as well as 5 adjacent folds are used as unlabeled training examples, and finally, the remaining 4 folds are used as testing examples. In this way, in each round, 60% of the data is used for training and remaining 40% of the data is used for testing. Each fold is used exactly once for the selection of labeled data, five times it is included as unlabeled data and four times it is used as a part of the testing set.

The base classifier used in co-training algorithm is Naive Bayes (NB). This classifier was chosen both for its speed (which is an important factor due to the complexity of *RSSalg*) and ability to handle missing data.

The problem of movie rating prediction can be highly imbalanced – before watching a movie users usually refer to the plot description, producer, cast and other factors they find important in order to see whether the movie appeals to them [9]. The consequence is that users will generally watch and rate movies that they like. Thus, as the measure of performance we use both accuracy and micro and macro *f*-measure [13].

In our experiment setting we assume the scenario where we already have a number of users in the system and the new user is joining the group. New users have a very small number of ratings. In our experiments we make them rate only 3 movies that they *like* and 3 movies that they *dislike*, i.e. the small initial training set *L* for co-training consists of 3 positive and 3 negative examples.

The number of examples labeled by co-training inner classifiers in each iteration are chosen proportional to the class distribution in the dataset as suggested in [6]. The size of the unlabeled pool is 50. We run co-training algorithm until we label all unlabeled data. The number of different random splits used in *RSSalg* is 100. All these parameters were empirically chosen.

For the *User view* we form for each user we use the top 50 users in terms of number of rated movies. Ideally, we would want to utilize all users in the system and perhaps apply some dimensionality reduction technique, or choose the subset of users most strongly correlated (positively or negatively) to the new user. However, due to the small number of ratings we have for the new user, we are unable to obtain a reliable similarity measure in order to determine those users. Thus, for now, in these initial experiments we simply use the top 50 users because of their high rating number which makes the *User view* less sparse.

For our experiment we have chosen 4 random users that we will treat as new users. The first two users ($User_1$ and $User_2$) belong to the group of top 50 users. The second two users ($User_3$ and $User_4$) do not belong to this group. As mentioned before, for each user we leave only 3 positive and 3 negative ratings (randomly chosen) and treat all other ratings as unlabeled/test data.

The reason we chose to take users from two different groups is to ensure that we are not getting good results only because the chosen users are highly correlated to the used set of top 50 users. By utilizing all ratings, for each "new" user we have calculated 50 most similar users. It turned out that most of the top similar users (more than 25) of $User_1$ and $User_2$ actually belong to the top 50 users, but both $User_3$ and $User_4$ only have one user that belong to both top rated and top similar groups.

Accuracy, macro and micro *F*-measure for $User_1$–$User_4$ are presented in Figures 1-4. Also, in Table 1 we give details about the accuracy achieved for $User_1$ by different co-training settings. Due to space limitations we omit details about *F*-measure, as well as other users, as they all display the same behavior.

The different settings we use (horizontal axes) are:

- *L*: NB classifier trained on the labeled portion of the dataset (6 examples);
- *All*: NB classifier trained on both labeled examples and unlabeled examples that are assigned the correct label. This is the goal performance we would like to achieve with co-training;
- *Natural*, *Random*, *MV*, *RSS* and *RSS_best* are the Co-Training settings introduced in section III.C.

The listed algorithms are tested using the different views introduced in section III.C.

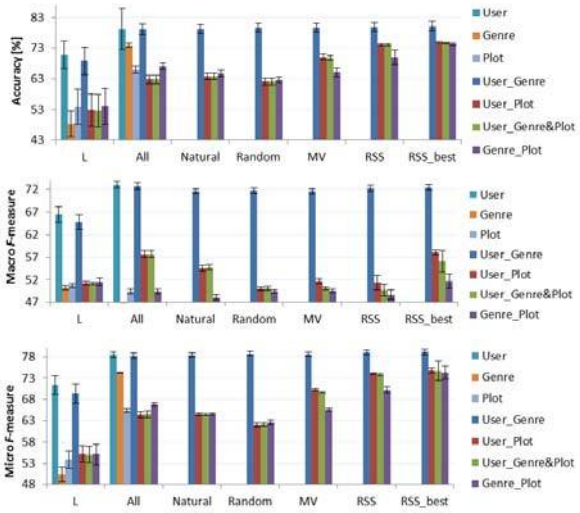

Figure 1 ***User*$_1$**: Accuracy, macro, and micro *F*-measure, respectively. The baseline accuracy is 74.0%, baseline macro F-measure is 42.7% and baseline micro *F*-measure is 74.6%. The number of annotated examples used in *All* is 541, and in *L* and co-training settings is 6.

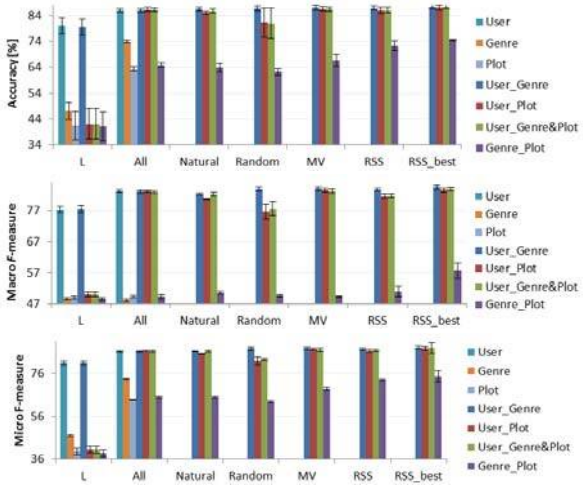

Figure 2 ***User*$_2$**: Accuracy, macro and micro *F*-measure, respectively. The baseline accuracy is 74.6%, baseline macro *F*-measure is 42.7% and baseline micro *F*-measure is 74.4%. The number of annotated examples used in *All* is 576, and in *L* and co-training settings is 6.

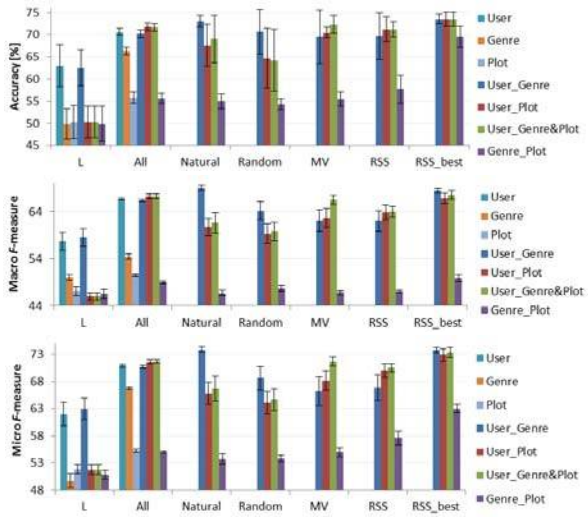

Figure 3 ***User*$_3$**: Accuracy, macro and micro *F*-measure, respectively. The baseline accuracy is 67.4%, baseline macro *F*-measure is 40.2% and baseline micro *F*-measure is 67.2%. The number of annotated examples used in *All* is 256, and in *L* and co-training settings is 6

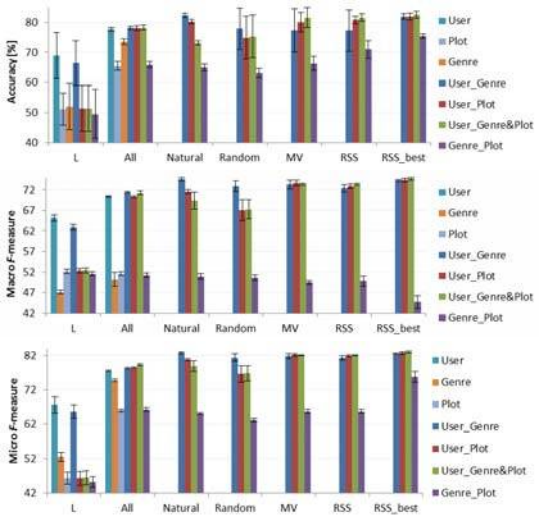

Figure 4 ***User*$_4$**: Accuracy, macro and micro *F*-measure, respectively. The baseline accuracy is 75.9%, baseline macro *F*-measure is 43.1% and baseline micro *F*-measure is 75.9%. The number of annotated examples used in *All* is 249, and in *L* and co-training settings is 6.

TABLE I.
USER$_1$: ACCURACY AND STANDARD DEVIATION FOR DIFFERENT ALGORITHMS/FEATURE SPLIT. THE SIZE OF THE ANNOTATED SET USED IN *ALL* IS 541, AND THE SIZE OF ANNOTATED SET USED FOR L AND CO-TRAINING SETTINGS IS 6

	User	***Genre***	***Plot***	***User_Genre***	***User_Plot***	***User_Genre&Plot***	***Genre_Plot***
L	74.7±8.2	48.3±8.4	53.9±11.5	74.5±8.1	62.2±15.7	62.1±15.7	54.1±11.6
All	79.9±3.2	73.9±1.3	66.0±2.4	79.7±3.2	79.9±2.8	79.9±3.0	67.1±2.2
Natural				80.7±2.9	72.3±16.4	80.4±3.2	64.7±2.2
Random				80.2±3.7	71.3±16.7	71.3±16.9	62.7±2.0
MV				80.4±3.1	76.4±11.2	75.8±12.7	65.1±3.0
RSS				80.3±3.3	80.0±4.2	79.8±3.8	70.0±4.8
RSS_best				80.9±3.2	81.0±3.9	81.0±3.8	74.3±0.5

Figures 1-4 and Table 1 show that the behavior of our algorithms is similar for all users. We can draw the following conclusions:

- Out of all single-views, in the terms of accuracy and macro and micro *F*-measure different settings have achieved using that view, *User view* is the strongest. This is not surprising as CF algorithms are shown to generally outperform content-based algorithms, but also due to the fact that we are using very basic features for the content classifier which we tend to improve in the future.
- The weakest single view is the *Plot view*. We have constructed the *Plot view* based on IMDB plot descriptions and as the authors in [9] notice – the storyline in IMDB is generally short and the resulting dataset is very sparse, which is probably the reason for the poor performance of the *Plot view*.
- For NB single-view performance (*L* and *All*), combining *User view* with other views (i.e. *User_Genre*, *User_Plot* and *User_Genre&Plot*) only slightly (if at all) improves the performance of the User view.
- The combination of *Plot* and *Genre* views (i.e. *Genre_Plot*) is the weakest one, in some cases even weaker than *Genre* and *Plot* views alone (for both *L* and *All*). Even if we had labels for all training data (*All*), *Genre_Plot* combination is worse than *User* trained on the labeled data alone (*L*). This is not surprising as Authors in [9] state that there is a high correlation between the movie genre and words in the plot description, and these highly correlated attributes are probably the reason for poor performance.

However, although features from *Genre* and *Plot* seem useless when compared to *User view* in a single-view setting, we can see from the results that they are very useful in a multi-view setting:

- For all view combinations, *Natural* was able to greatly improve the performance of the weak initial classifier (*L*). Its performance is substantially better from both the combined version of the views and single views (e.g. *Natural* applied to *User_Genre* split is better than *L* on *User_Genre*, *L* on *Genre alone* and *L* on *User* alone). Also, for all multi-view settings except *Genre_Plot*, the performance of *Natural* is in the rank of performance of *All* setting, meaning that by applying co-training on a very few labeled examples we have succeeded to achieve the performance we could achieve if we had a large number of labeled examples. For *Genre_Plot* the performance of *Natural* is slightly worse than *All* setting for the same view, but it is still able to greatly improve the performance of the weak initial classifier (*L* setting on the *Genre_Plot* view).
- As expected, for all settings *Random* achieves worse performance than *Natural*. *MV* is better than *Random* and *Natural* and even slightly surpasses the performance of *All* (it achieves a slightly better accuracy, but also better micro and macro F-measure). *RSS* is slightly worse than *MV*, but it is in the rank of *All* setting. Finally, the

RSS_best setting outperforms all settings, but as it is optimized on the test set, it can only be seen as the upper bound for the performance of *RSS* algorithm [7].

- As for the views, the best combinations are *User_Genre* and *User_Genre&Plot* (whose performances are approximately the same).
- Not surprisingly, the *Genre_Plot* multi-view setting has the worst performance of the multi-view settings. *RSS* and *RSS_best* are able to boost the performance of the weak initial classifier beyond the performance of *All* setting for the *Genre_Plot* view combination. This is consistent with findings in [7] – *RSSalg* has the best performance if the features are highly redundant, and in *Genre_Plot* view where we have correlated features [7].
- The performance of *RSS_best* in this case is in the rank of *All* setting for the single *Genre view*. However, in some cases (e.g. $User_2$), this performance can still be worse than the performance *L* setting has on the *User view*. However, it should be noted that this combination can still be very useful. Consider the situation where the new user has only rated the movies that none of the other users has rated. In this situation, applying co-training, or, better, *RSS* algorithm with this view combination can significantly boost the performance of the initial classifier. Finally, we should note that these are only the initial experiments. In the future we plan to enhance the content classifier, e.g. by including semantic [9].

V. Conclusion

In this paper we address the new-user problem in a recommender system, i.e. the situation where, due to the lack of user's rating history, the recommender system is unable to give quality personalized predictions. We have posed the recommendation problem as a classification problem. For each user, we build a separate model which, given an item unrated by user, predicts whether the user will like or dislike the item. In order to alleviate the new-user problem, we propose a multi-view hybrid recommendation system that uses other users' ratings as a first view and item description data as the second view. We apply our algorithm to the problem of movie recommendation and use the popular *MovieLens* dataset. As item description we use movie genre and plot description. We have tested several co-training settings using different co-training algorithms (classic co-training [6] and RSSalg [7]) and different view combinations. In all settings, by applying co-training we were able to improve the weak initial classifier (a supervised algorithm trained on the labeled version of the data) and even achieve the performance a supervised classifier would have if we had labels for all training data (both labeled and unlabeled). The best performing settings in our experiments were both co-training and RSSalg applied with users' ratings as the first view and the combination of genre and plot features as the second view, or users' ratings as the first view and just genre features as the second view. We also show that using a purely content predictor constructed with plot and genre features can be very useful in combination with co-training if there are no available ratings from other users.

In the results presented here, we showed that by starting from just 6 rated items, by employing co-training, we can achieve the performance a supervised system would have given a huge rating history. The results presented in this paper are just preliminary experiments performed in order to get a general idea whether a hybrid multi-view system we proposed could address the new-user problem.

There are many ways to extend this work in the future. First, we plan to enrich the content-based predictor used as the second view in our co-training based framework with other kinds of information, such as those used in [9]. In the experiments presented here, we have used only a subset of top 50 users in the terms of number of given ratings. We intend to experiment by utilizing all available user ratings. A task for the future is also to see whether our framework can also be applied to address the new-item problem.

Acknowledgment

Results presented in this paper are part of the research conducted within the Grant No. III-47003 financed by the Ministry of Education and Science of the Republic of Serbia.

References

[1] P. Resnick, and H. Varian, "Recommender Systems," *Communications of the ACM 40(3)*, 56–58, 1997.

[2] G. Linden, B. Smith, and J. York, „Amazon.com Recommendations: Item-to-Item Collaborative Filtering," *IEEE Internet Computing* 7(1), 76–80, 2003.

[3] Y. Chen, M. Harper, J. Konstan, and X. Li, „Social Comparisons and Contributions to Online Communities: A Field Experiment on MovieLens," *.American Economic Review* 100(4), 2010.

[4] Y. Wang, S.C. Chan, and G. Ngai, "Applicability of Demographic Recommender System to Tourist Attractions: A Case Study on Trip Advisor," *Web Intelligence/IAT Workshops* pp. 97-101, 2012.

[5] G. Adomavicius, and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. on Knowledge and Data Engineering*, vol. 17, pp. 734-749, June 2005.

[6] A. Blum, and T. Mitchell, "Combining labeled and unlabeled data with co-training," *Proc. Eleventh Annual Conference on Computational Learning theory COLT'98*, ACM, pp 92-100, 1998.

[7] J. Slivka, A. Kovačević, and Z. Konjović, "Combining co-training with ensemble learning for application on single-view natural language datasets," *Acta Polytechnica Hungarica*, Vol. 10, No 2, pp. 133-152, 2012.

[8] D. Billsus, and M. Pazzani, "Learning collaborative information filters," *Int'l Conference on Machine Learning*, Morgan Kaufmann Publishers, 1998.

[9] W. Qu, K-S. Song, Y-F. Zhang, S. Feng, D-L. Wang, and G. Yu, "A Novel Approach Based on Multi-View Content Analysis and Semi-Supervised Enrichment for Movie Recommendation," *Journal of Computer Science and Technology* 28(5): 776-787, September 2013.

[10] J. Delgado and N. Ishii, "Formal Models for Learning of User Preferences, a Preliminary Report," *Proc. Int'l Joint Conf. on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden, July, 1999.

[11] R. Ghani, and A. Fano, "Building recommender systems using a knowledge base of product semantics," *Proc. Workshop on Recommendation and Personalization in E-Commerce, at the 2nd Int'l Conf. on Adaptive Hypermedia and Adaptive Web Based Systems*, Malaga, Spain, May 2002.

[12] D. Jannach, L. Lerche, F. Gedikli, and G. Bonnin, "What recommenders recommend - An analysis of accuracy, popularity, and sales diversity effects," *21st Int'l Conf. User Modeling, Adaptation and Personalization (UMAP 2013),* Rome, Italy, 2013.

[13] M. Sokolova, and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manage*, 45(4): 427-437, 2009.