Clustering to Deal with the New User Problem

Christos Bouras
Computer Engineering and Informatics Department,
University of Patras and
Computer Technology Institute and Press "Diophantus"
Patras, Greece
bouras@cti.gr

Vassilis Tsogkas
Computer Engineering and Informatics Department,
University of Patras
Patras, Greece
tsogkas@ceid.upatras.gr

Abstract— **Collaborative filtering (CF) techniques attempt to alleviate information overload by identifying which items a user will find interesting to browse. It focuses on identification of other users with similar tastes and usage of their opinions in order to recommend items. Commonly, however, CF suffers from the so-called new user problem which occurs when a new user is added to the system and there is not enough information to make a good suggestion. The system has to acquire some data about the new user in order to start making personalized recommendations. In this paper, we present a novel algorithm that combines previously acquired knowledge from article and user clustering in order to quickly determine the new user's interests. We attempt to address the new user problem by providing a personalized strategy for prompting the user with articles to rate. Our approach makes use of hypernyms extracted from the WordNet database and proves to be converging fast to the actual user interests based on minimal user ratings which are provided during the registration process.**

Keywords New user problem; collaborative filtering; clustering; W-kmeans; Personalized strategy

I. INTRODUCTION

Each day, more and more books, journal articles, web pages, and movies are available online. While available information is growing in volumes, we quickly become overwhelmed and seek assistance in finding the most interesting, valuable, or entertaining items on which we should spend our scarce time. Historically, humans have adapted well to pieces of information and have developed an excellent filtering ability to make quick judgments.

The technologies that are commonly used to address the previously mentioned information overload challenges are basically three. Each one focuses primarily on a particular set of tasks or questions: a) information retrieval (IR), which focuses on tasks involving fulfilling ephemeral interest queries, such as finding the articles related to president Obama, b) information filtering (IF), which focuses on tasks involving classifying streams of new content into categories, such as finding any newly released articles regarding the economic situation in Greece, or any newly released movies without an English-language soundtrack or subtitles (to reject), and c) collaborative filtering (CF), which focuses on two important questions: which items (from a set or overall) should be proposed to a user, and how appealing will these particular items be for the user. Each of the above technologies has a specific approach in producing an effective recommender system.

Collaborative filtering is a methodology by which users co-operate in order to determine what is interesting or useful from a large set of items. The intuition behind this approach is that since people prefer only to look at relevant, interesting content, one way to filter out the irrelevant data is to collaborate with other users and consider what they believe to be relevant. The first collaborative filtering system was Tapestry [1] which was an attempt to select informative email messages from mailing lists. Grouplens [2] was another CF system that was able to automatically select the group of trusted users for the current one. MovieLens, being the latest system from the GroupLens group, attempts to make rating predictions and recommend movies based on collaborative filtering. Nowadays, there are many more CF systems online (Amazon, Google News, YouTube, Netflix, etc.) most of which generate recommendations to the user, based on those items that have the highest predicted rating among those that the user has not yet rated. The previous technique makes CF and recommender systems very similar.

A. The new user problem

A common problem that all CF filtering systems suffer from is the cold start problem. It consists of a family of three related problems: a) the new item problem, where a new item is introduced to the community and since it has no rating, the system cannot recommend it to any user, b) the new user problem, where a new user enters the system and there are no ratings made yet by her – hence there can be no generated system predictions for her, and c) the new system problem, where we have a new system for which there are no ratings of any users. In this paper, we will be focusing on the new user problem.

Previous approaches to the new user problem have mostly focused on metadata and user-prompting. The metadata about items can be used to generate recommendations by content-based recommender systems like in [3] or in a hybrid fashion with ratings-based system as the ratings come into the system like in [4]. Filterbots [5] are another approach in which pseudo-users and items are algorithmically created in an attempt to provide base ratings to the system such that no user or item would be left without rating. These agents, as shown in [14], can potentially be performing better when working in tandem with CF techniques and in particular, the CF engine is what matters the most in this combinatory scenario. Other methods that utilize demographic data available to the system have also been proposed, but gathering such data often conflicts with privacy issues.

Recommender systems have also been used to help tackle the new user problem. Some approaches, like [16], create user categories where new users are quickly assigned by using a set of pre-determined questions. These approaches jump-start the system by using demographic or model based attributes. Even though domain-limited, they can potentially produce accurate results.

B. *User prompting*

Another method for dealing with the new user problem is to explicitly ask the users to provide ratings for items (i.e. news articles in our case). The scheme is pretty basic: when the new user enters the system, she is presented with items to rate, which are not really recommendations, but are rather selected in order to gather as much information about her profile as possible. As she continues providing ratings for the selected items, the system decides whether to continue this process improving the user's profile or halt it. Large questionnaires come with a cost though: users are easily disturbed and might give up the process if it's taking too long or if they feel that the requested data conflict with their privacy. When this process ends, the system, having a basic knowledge of the user's appetite, starts recommending items and monitors her actions forming a feedback loop for profile updates.

This prompting approach was introduced in [6] where the ordering of items by the variance and entropy was investigated. Methodologies regarding user prompting can be divided into non-personalized and personalized [11]. Non-personalized methodologies for user prompting include: a) the popularity method, where items are ordered by the number of ratings that they have been given by all users, b) the entropy and its variations methods that rely on the fact that certain items can yield more information about a user's likes than others, c) the greedy method, where the next item is chosen from those that the user is able to rate, such that the prediction error for her test set is minimized (clearly this method could not be used in practice as it requires knowing not only what each person is able to rate but also the actual ratings), d) other people's greedy and variations, where the items that will be presented to the user are chosen from the top-n lists of other users. Personalized methodologies on the other hand take into consideration the responses that the user has given to items already presented. Some non-personalized methodologies are: a) naïve Bayes, where by knowing whether a user is able to rate an item we can work out the naïve Bayes probability of a user being able to rate the other items, b) perturbed other people's greedy and variations which combine other people's greedy with naïve Bayes.

In [7] the authors presented more methods for improving the order of items attempting also personalized orderings. Recently a new method for a non-personalized ordering of items was presented in [8] and a personalized one in [9]. In [10] the authors, using a prediction method which was a variant of matrix factorization, showed that more accurate predictions can be made when the user has provided minimal ratings than when the system uses the metadata of the items in order to generate predictions. There are two important aspects for the user prompting approach: a) which items to select and b) in which order to present them to the user.

There have been many approaches regarding the process of selecting which items to present next to the user during the prompting phase. Certain trade-offs should also be considered, like the effort the user has to put into and the satisfaction she will get by the registration process as a whole. Moreover, the recommendation accuracy, i.e. how well the recommendations presented to the user really are, is of great significance. Taking the above into consideration, there have roughly been 5 main strategies for selecting which items to present to the user during the prompting phase: random, popularity, pure entropy, balanced and personalized [15].

In random strategies, the items that are to be presented are chosen randomly with a uniform probability over the universe of items. If the distribution of ratings is uniform, they have the advantage of covering the entire universe of items. In the popularity based strategies, the items are sorted in descending order based on their number of ratings. Even though easy to compute, these strategies overly promote items that have been widely rated and may carry little information. In pure entropy strategies the users are asked about items which give the most information for each rating. Generally, an item that has some people that disliked it and some other who liked it may tell us more than an item which everybody liked. In balanced strategies, a combination of the popularity and entropy strategies is used. This can usually be in the form of Popularity*entropy or Log Popularity*Entropy. This approach, using Bayes theorem, silently assumes that popularity and entropy are independent which is not always correct. Lastly, in personalized strategies the suggestions are adapted to the user preferences via a feedback loop. Frequently, the item by item strategy is used in which, at first, items are presented to the user by any other strategy until a rating is picked up. Following that, a recommender, based on some similarity measure, finds similar items for user suggestions based on what the user previously rated.

In our work we are focusing on a personalized methodology for user prompting, similar to the item by item strategy. Our approach exploits clustering, and in particular our W-kmeans clustering algorithm [12], in order to select which news articles to pick next for rating by the user. Our work explores both item (i.e. article) and user clustering in order to effectively select articles for rating and thus quickly and reliably converge to the actual user's preferences. We also try to determine how well our approach deals with the new user problem compared with the basic 5 strategies that were previously described.

The rest of the paper is structured as follows: Section 2 presents our system as a whole and the flow of information within it. In Section 3 we present our personalized algorithmic strategy for dealing with the new user problem. In Section 4 we present our experimentation as well as its results, while in Section 5 we outline the conclusions of this work as well as some areas that are worth considering for future research.

II. Flow of Information

Fig. 1 depicts the flow of information that is followed in the suggested approach [13]. Initially, at its input stage, our system fetches news articles generated by news portals from around the Web. This is an offline procedure and once articles as well as metadata information are fetched, they are stored in the centralized database from where they are picked up by the procedures that follow. A key process of the system as a whole, probably as important as the clustering algorithm that follows it, is text preprocessing on the fetched article's content, that results to the extraction of the keywords each article consists of. Analyzed in [13], keyword extraction handles the cleaning of articles, the extraction of the nouns, the stemming as well as the stopword removal process.

Keyword extraction then applies several heuristics to come up with a weighting scheme that appropriately weights the keywords of each article based on information about the rest of the documents in our database. Pruning of words, appearing with low frequency throughout the corpus, which are unlikely to appear in more than a small number of articles, comes next. Keyword extraction, utilizing the vector space model, generates the term-frequency vector, describing each article as a 'bag of words' (words – frequencies) to the key information retrieval techniques that follow: article categorization, summarization and clustering. We are enhancing the efficiency of this 'bag of words' with the use of an external database, WordNet. The above characteristics of our system give its content-based nature. This enhanced feature list feeds the k-means clustering procedure that follows [12]. It is important to note, however, that the clustering process is independent from the rest of the steps, meaning that it can easily be replaced by any other clustering approach.

Following the core IR tasks of our mechanism, the personalization algorithm takes place. The personalization module can easily adapt to subtle user preference changes. Those changes, as expressed by the user's browsing behavior, are detected and continuously adjust her profile. The algorithm uses a variety of user-related information in order to filter the results presented to the user. Furthermore, it takes into account in a weighted manner the information originating from the previous levels regarding the summarization/categorization and news/user clustering steps.

User profiles from multiple users and timeframes are then clustered using the W-kmeans algorithm forming profile clusters. W-kmeans is a novel approach that extends the standard k-means algorithm by using the external knowledge from WordNet hypernyms for enriching the "bag of words" used prior to the clustering process. The W-kmeans algorithm enhances the user profiles with hypernyms deducted from the WordNet database, using a heuristic manner.

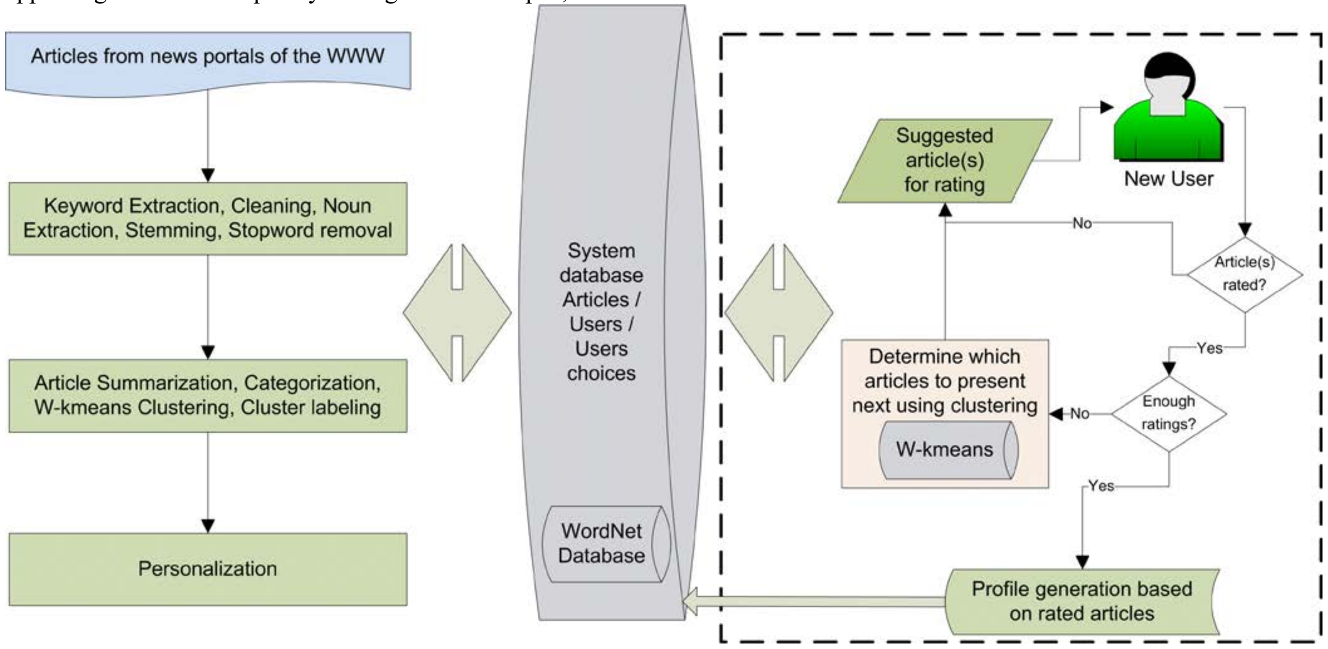
Articles from news portals of the WWW
Keyword Extraction, Cleaning, Noun Extraction, Stemming, Stopword removal
Article Summarization, Categorization, W-kmeans Clustering, Cluster labeling
Personalization
System database Articles / Users / Users choices
WordNet Database
Suggested article(s) for rating
New User
No
Article(s) rated?
Yes
Determine which articles to present next using clustering
W-kmeans
No
Enough ratings?
Yes
Profile generation based on rated articles

Figure 1 Flow of Information

Those profile clusters are used at the recommendation stage in order to enhance the system's usage experience by providing more adapted results to users revisiting the site. When a user comes back her clustered profile is recalled. Articles matching her profile are extracted and are considered for user recommendations.

The dashed component of Fig. 1 captures the high level approach that is proposed in this work for dealing with the new user problem. When a new user enters the system, she enters a priming phase (user-prompting) in which a series of articles are suggested to her as rating candidates. Our goal is twofold: we want the total number of presented articles to be minimized, while also allowing the system to gain as much information as possible from the rated articles. Determining which articles to select for the user prompting phase and in which order are the main aspects that the current work focuses on.

Initially, we suggest articles for user rating based on a given strategy: even though one would expect that this strategy would be of significance, this is not really the case as shown in [15]. Thus, we simply select articles coming from the most rated list that resides in the database. While the suggested articles are not rated by the user, we continue the suggestions based on this list. Once the user has rated one article, we utilize our clustering data in order to find out and suggest for subsequent rating, articles that a) belong to the same article cluster as the rated one or b) are chosen by users that have rated the selected article(s) likewise before. The above procedure continues until enough user ratings are gathered. Once this phase is complete, the user profile is boot-strapped and the system goes into its normal recommendation state.

III. Algorithmic Approach

In this section we will be describing the various algorithmic steps that are followed in our approach. Algorithm 1 presents the procedure that is used for gathering ratings from a new user who is visiting and registering to the system for the first time. Algorithm 2 outlines the steps used to recover articles based on either article or user clustering information. We don't get into much detail on how each function works, but the names should be self-explanatory.

```
Algorithm harvest_user_ratings
Input: NULL
Output: user_ratings[] //rated articles by the user
   rated_article = NULL // first rated article A1
   article_cluster = NULL
   articles_next [] = NULL
   rated_articles[] = NULL
   while (!rated_article)
      rated_article = rate(present_next_most_rated_article())
      //continue presenting from the L1 list until user has rated 1 article
   user_ratings[] += rated_article // Article A1 is rated with score S1
   article_cluster = find_article_cluster(rated_article)
   articles_next[] = find_most_rated_articles(article_cluster, M)
   // articles_next[] is now list L2 containing M items
   while (has_next(articles_next))
      rated_articles[] = rate(present_next_article(articles_next))
   if (!rated_articles[]) // user hasn't rated any of the M articles from L1
      articles_next[]=find_most_rated_articles_from_user_clusters (rated_article,M)
      // articles_next[] is now list L3
      rated_articles[] = rate(articles_next[])
      user_ratings [] += rated_articles[]
     GOTO: T // Continue possible suggestions from user clustering
   else //user has rated some of the M articles
      user_ratings [] += rated_articles[]
T:      while (user_ratings.size() < Rmin) // do we have enough ratings?
          articles_next[]=find_most_rated_articles_from_user_clusters (rated_articles,M)
          // articles_next[] is now list L4
          rated_articles[] = rate(articles_next[])
          user_ratings [] += rated_articles[]
   return user_ratings[]

Function rate
Input: articles[]
// Presents for rating the selected article(s) and returns the rating scores
// or null if the article wasn't rated
  rated_articles[]=NULL
Output: rated_articles[]

Function find_article_cluster
Input: rated_article
// Recovers the cluster that the article belongs to
Output: article_cluster
```

Algorithm 1. Determining which articles to present to the user

```
Algorithm find_most_rated_articles
Input: cluster, M
// Recovers the M most rated articles that belong to this cluster
// Uses article clustering results from the database
// cluster can be either an article or a user cluster
Output: articles[M]

Algorithm find_most_rated_articles_from_user_clusters
Input: article / articles[], size M
// Recovers articles from the user clusters which contain users who have
// previously rated the specified article(s). Uses user clustering results
Output: rated_articles[]
    rated_articles[]=NULL
    clusters[]= find_user_clusters(article)
    //find clusters of users who have rated article(s)
    for each cluster in clusters[]
        rated_articles[]+=find_most_rated_articles (cluster, M)
    return rated_articles[]
```

Algorithm 2. Recovering articles based on article or user clusters

In our work, we use an item by item personalized strategy, as the one described in [15], in order to select articles as rating candidates for the new user. Initially, when a user enters the prompting phase, we present her with articles one by one from the most rated (popular) list of articles that reside in the system's database. Let's call this initial list: L1. The presentation of articles continues until an article, say A1, is rated by the user with score S1. We use this information in order to determine the cluster to which this article belongs. Then, we can suggest for user rating M of the most rated articles from this cluster, which are forming article list L2. Note that L2 contains articles based on article clustering information originating from the system's database. Choosing the correct value for M is a matter of experimentation since there is a specific trade-off for it, which we will try to briefly describe. Large M values give many similar articles, i.e. from the specific article's cluster, so that a hit there, regarding the user's preferences, will probably harvest many user ratings. However if the rated article A1 does not convey entirely the user's appeals, many articles that with high probability won't get rated will be presented without an easy way for her to backtrack. This can have a negative effect on the system's performance and will probably also cause user frustration. On the other hand, small values of M might lead to a similar negative impact via a different path: a user would expect from the recommendation system to catch up her preferences quickly and not backtrack to articles she has no interest about. In a nutshell, we don't want to overload the user with articles from a single cluster, but still, we want to determine relatively fast if articles belonging to that cluster are really interesting to the user. Additionally, we want to cover as broadly as we can the related clusters that might capture user ratings, thus a small to medium value for M should be more reasonable.

As our algorithm proceeds, if no rating is given on any of those M articles of L2, we seek for the user clusters which contain users who have previously rated the item Ai with score Si. Using these user clusters we can form an article list, say L3, which consists of M * number of clusters of the most rated articles. Again we choose to keep M articles from each of these user clusters and as before the same tradeoffs apply on the value of M. The L3 list, containing user clustering information is subsequently suggested to the user and any possible article ratings that are gathered from the user are used to recreate the L3 list in a similar fashion. This process forms a loop until the total number of ratings reaches our defined threshold, say Rmin.

On the contrary, if the user has rated at least one of the M articles of the L2 list, we seek for the user cluster(s) that contain most of the previously rated articles and again, we select the (M * number of clusters) of the most rated articles (list L4). Although resembling each other, the L3 and L4 lists are not the same: the difference is that L3 is based entirely on user clustering while L4 is instantiated via article clustering first and enhanced later on via user clustering using collaborative knowledge that resides in the system's database. We have selected this article/user clustering combination based on our previous experimental results [12]. The aforementioned approach continues until at least Rmim user ratings are gathered.

Once the needed user ratings are harvested, the registration process ends and the user can now browse through the personalized recommendations that the system provides. As explained in Section 2, the personalization improves the quality of the user-suggested articles based on the continuous feedback that the user provides via his choices.

IV. Experiments and Results

For our experimental procedure we built a dataset using a snapshot of our system's database, PeRSSonal [13]. The reason that we didn't select to work with other datasets (like NetFlix or MovieLens), is due to the clustering capabilities of our system: W-kmeans is enabled and already applied on our dataset. During our evaluation, we firstly eliminated users who had fewer than 50 recorded ratings. This left us with 60 users who had rated 2,055 articles with over 10,000 ratings. Using a cutoff of 50 ratings is significant: we want to prevent users who haven't used extensively our system from affecting the evaluation process. In general, we need many ratings for each user in order to have a good sample of articles that they were able to rate which effectively constitutes their preferences. Since a "new" user in the experimentation that follows is practically each one of the 60 users we previously mentioned, for each run we withheld all of the user's ratings from the system. As we presented articles based on each of the strategies that we are evaluating, users "rated" the articles they have "viewed", i.e. the ones for which we had ratings in the database. This means that if an article that is presented to the user was found as rated or viewed in our database we consider it as a successful proposal for user rating. Regarding the particular scores, the rating score found in the database is used as the rating score she would give during the prompting phase. If the article was found in the viewed list, we consider this with the highest rating score, i.e. 5. For our first and second experimentation set, we stopped presenting articles once we got the required number of ratings, which for our experimentation was set to be Rmin = 20.

For our first experiment, we tried to determine the best value of the parameter M described in Section 3. That is: the best amount of articles we should present to the user, which belong to a certain cluster (either article or user cluster) after one or more article ratings are harvested from the user. In order to perform this evaluation, we deployed one of the most widely used evaluation metrics for predicting performance of recommender systems, which is the Mean Absolute Error (MAE). We use the MAE, for expressing the average absolute deviation between the predicted and the actual user ratings. MAE can be computed using formula (1).

$$MAE = \frac{\sum r'(u,i) \in R' \mid r(u,i) - r'(u,i) \mid}{\mid R' \mid} \qquad (1)$$

where $\mathrm{r(u,i)} \in [1,5]$ is the actual rating (recorded in the database) of user u for article i and $\mathrm{r'(u,i)} \in [1,5]$ the predicted / recommended preference for user u of articles belonging to the space of proposed articles, R'. For this experiment we used an increasing number of values for M on each run, starting at M=1 and ending at M=50. The results are presented in Fig. 2.

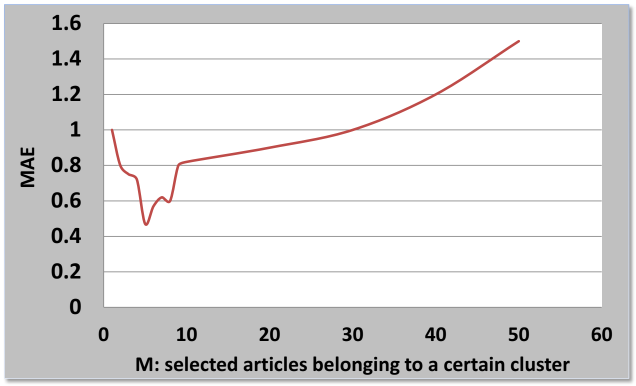

Figure 2 MAE scores for various numbers of M

From the graph of Fig. 2 we can pinpoint that the best value of the parameter M in terms of MAE is M=5. From a natural point of view this means that selecting 5 articles from the articles or user clusters is best for forming the M2, M3 or M4 lists. Executions with lower M values suffered from few article suggestions coming from article and user clusters, something that was leading to low performance and longer user prompting times. We expect that on an online experiment with many users, the performance would have been even worse, counting in also the user frustration which is to be expected when the prompting phase takes too long. Evidently, values for M > 10 also suffered from poor performance. This can be explained by the fact that when we are using too many articles from article and user clusters, the users have a hard time backtracking out of a cluster if they need to when going through the prompting phase. For our subsequent experiments, we are using M=5.

For our second experiment, we used each of the user prompting strategies described in Section 1: entropy, random, popularity, balanced, personalized item by item, as well as our proposed clustering-based strategy, in order to select which articles to present to the new user. Once the prompting phase completed for each strategy, we counted the number of articles that the user had to view until the Rmin = 20 ratings were harvested. We need to stress out that the fewer the articles that we had to present to the user, the better, given the fact that we are saving user effort. Also note that for the personalized item by item strategy, we used the popularity based approach for presenting the initial articles until a rating is accomplished by the user. This is similar to the proposed clustering-based methodology that we follow, except of course that we are also exploiting the clustering information. After that, we found out which articles are close to the rated one by utilizing the cosine similarity as a measure. The articles' similarity is calculated by using each article's keywords as explained in [13]. The similar articles are next used as suggestions for users to rate.

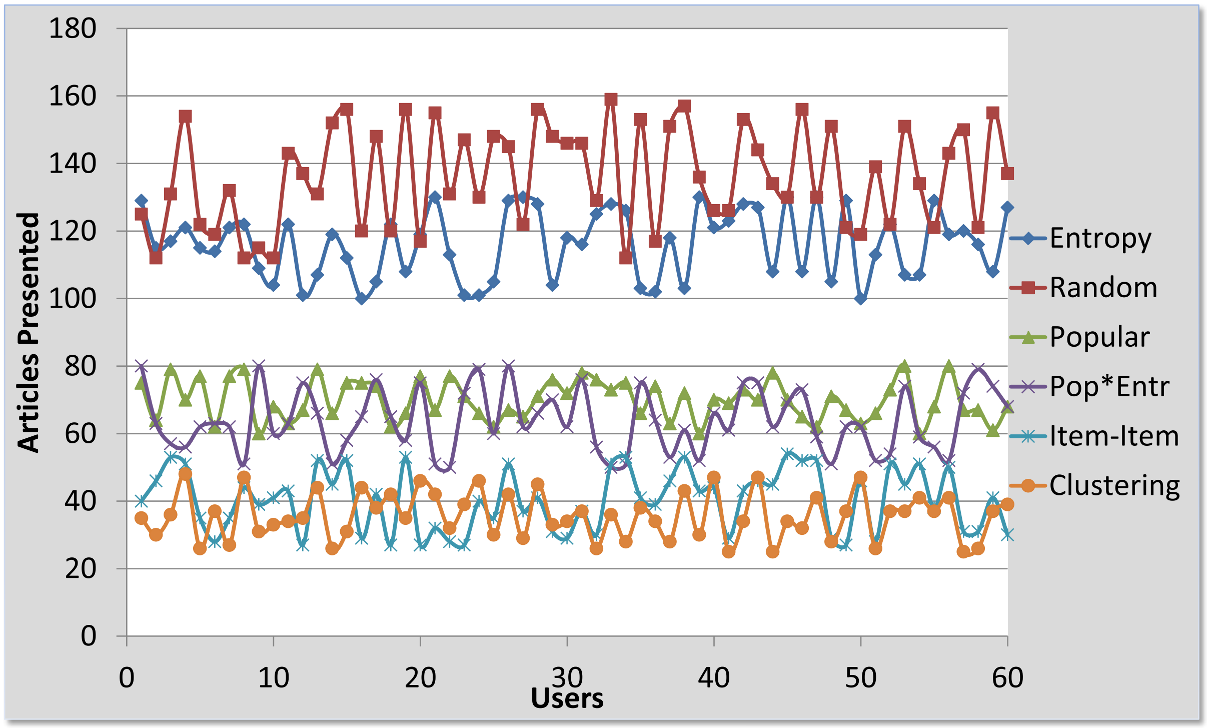

Figure 3 Required number of articles per strategy during the prompting phase

From the graphs depicted in Fig. 3, we can observe that our clustering-based article selection approach clearly outperforms all other strategies. We see that the random strategy required an average of 135 articles to be presented to the user before 20 ratings were harvested. This result is expected due to the random nature of this strategy and the fact that the user ratings cannot be considered as uniform throughout the dataset: each user is expected to have rated articles that only appealed to her and only for particular domains. The same number (for the amount of articles) was 115 in the case of the entropy-based strategy and 70 for the popularity-based strategy. The results for the entropy strategy, though surprising, have a plausible explanation: this strategy promotes less popular articles. However, there is a straight correlation between popular articles and the chance that a new user would like a popular article. Thus by choosing less popular articles at most of the times, this strategy suffers from bad performance. The results were better for the balanced popularity*entropy strategy with 55 articles on average. The personalized item by item strategy, even though very promising with an average of 41 articles, couldn't match the average of 37.5 articles that our approach scored.

For our third experiment, we tried to determine the prediction accuracy of the proposed approach compared to the previously mentioned strategies. Again, we made use of the MAE metric. For determining the MAE scores of each strategy, we presented for user rating a total of 30, 50, 70 and 90 articles in 4 subsequent runs for each of the 6 strategies. The results, presented in Fig. 4, show the MAE variations of the different strategies as a function of the presented articles.

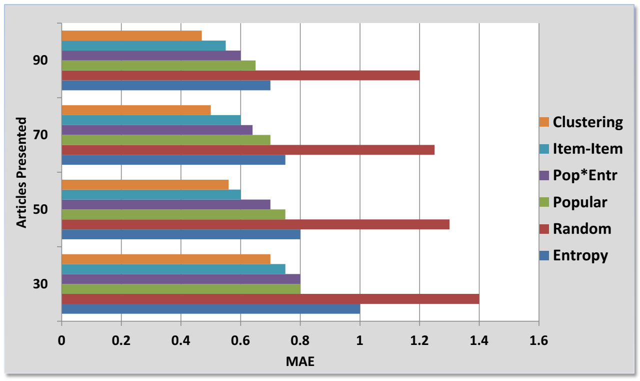

Figure 4 MAE scores for each of the prompting strategies for various number of presented articles

We can observe a MAE improvement as the number of presented articles increases, something that has a significant effect especially for the proposed clustering-based article selection strategy: as more and more articles are rated by the user, our approach can pick up better candidates for user rating by utilizing article and user clustering data from the database. Indeed the proposed approach gives the lowest MAE scores for each of the experiment's executions. Another result we can pick up from Fig. 3 is that the random strategy has the worst prediction accuracy, validating our observations on the first experimentation. We can also observe that the personalized item by item strategy is again, as in the second experiment, close to our proposed strategy.

V. Conclusions and Future Work

In this paper we have presented a personalized strategy for dealing with the new user problem via prompting, in the domain of news articles. We implemented and evaluated an algorithmic approach that makes use of article and user clustering information in order to decide upon which articles to present next for rating during the process of user registration. Our approach has similarities to the personalized one by one strategy, but the results have shown that it performs better than any of the most commonly proposed strategies for the problem of user prompting.

Our experimentation, based on offline system user ratings, showed that by using M=5 articles from each article or user cluster we get the lowest MAE scores. Using this result, we later determined that our approach requires, on average, 37.5 articles to be suggested for user rating in order to acquire 20 ratings; an amount of articles that was considerably less than any other techniques that similar systems use. Finally, we compared the MAE scores of the proposed technique with each of the entropy, random, popularity, balanced and personalized item by item strategies. Each experiment, executed over various amounts of articles has also shown that our methodology outperforms all of the afore-mentioned strategies.

Even though the above results are encouraging, they cannot be considered as conclusive, since they were done over an offline snapshot of the system's database based on already recorder user ratings. Thus, for the future we are considering a more large scale experimentation that will also include online data by more users registering through our system. We would also like to evaluate our approach with other databases too, such as the NetFlix and MovieLens datasets. Furthermore, we are working on enhancing our user prompting algorithm so that it includes more personalized information that can be harvested from the system. In particular, we are looking for an extension that will include the categorization as well as the summarization information which will enhance the personalization factor of our algorithm.

Acknowledgment

European Union
European Social Fund
OPERATIONAL PROGRAMME
EDUCATION AND LIFELONG LEARNING
investing in knowledge society
MINISTRY OF EDUCATION, LIFELONG LEARNING AND RELIGIOUS AFFAIRS
MANAGING AUTHORITY
Co- financed by Greece and the European Union
NSRF 2007-2013
EUROPEAN SOCIAL FUND

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

REFERENCES

[1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using collaborative filtering to weave an information tapestry,” Communications of the ACM, v.35 n.12, Dec. 1992, pp. 61-70.

[2] P Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “GroupLens: an open architecture for collaborative filtering of netnews,” In Proceedings of the 1994 ACM conference on Computer supported cooperative work, October 22-26, 1994, Chapel Hill, North Carolina, United States, pp. 175-186.

[3] M. Balabanovie and Y. Shoham. “Fab: Content-based collaborative recommendation,” Communications of the ACM, 40, 1997, pp. 66-72.

[4] K. – Y. Jung, D. Park, and J. Lee, “Hybrid collaborative filtering and content-based filtering for improved recommender system,” Computational Science- ICCS 2004, pp. 295 – 302.

[5] S. Park, D. Pennock, O. Madani, N. Good, and D. DeCoste, “Naïve filterbots dor robust cold-start recommendations,” In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 699 – 705.

[6] A. Kohrs, and B. Merialdo, “Improving Collaborative Filtering for new-users by Smart Object Selection,” In proceedings of International Conference on Media Features, 2001.

[7] A. Rashid, G. Karypis, and J. Riedl, “Learning preferences of new users in recommender systems: an information theoretic approach,” ACM SIGKDD Explorations Newsletter, 10(2), 2008, pp. 90-100.

[8] N. Golbandi, Y. Koren, and R. Lempel, “On bootstrapping recommender systems,” In Proceedings of the 19th ACM International Conference of Information and Knowledge Management, ACM, 2010, pp. 1805-1808.

[9] N. Golbandi, Y. Koren, and R. Lempel, “Adaptive bootstrapping of recommender systems using decision trees,” In Proceedings of the Forth ACM International Conference on Web Search and Data Mining, 2011, pp. 595-604.

[10] I. Pilaszy, and D. Tikk, “Recommending new movies: even a few ratings are more valuable than metadata,” In Proceedings of the Third ACM Conference on Recommender Systems, 2009, pp. 93-100.

[11] M. Crane, “The New User Problem in Collaborative Filtering,” Thesis for the degree of Master of Science, Department of Computer Science, University of Otago, 2011.

[12] C. Bouras, and V. Tsogkas, “Clustering User Preferences Using W-kmeans,” In Seventh International Conference on Signal-Image Technology and Internet-Based Systems (SITIS), 2011, pp. 75-82.

[13] C. Bouras, V. Poulopoulos, and V. Tsogkas, “PeRSSonal's core functionality evaluation: Enhancing text labeling through personalized summaries,” Data and Knowledge Engineering Journal, Elsevier Science, Vol. 64, Issue 1, 2008, pp. 330 – 345.

[14] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl, “Combining collaborative filtering with personal agents for better recommendations,” In Proccedings of the 16th international conference on Artificial intelligence and the 11th Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, July 18-22, 1999, Orlando, Florida, United States, pp.439-446.

[15] A. M. Rashid, A. Istvan, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl, “Getting to know you: learning new user preferences in recommender systems,” In Proceedings of the 7th international conference on Intelligent user interfaces, January 13-16, 2002, San Francisco, California, USA, pp. 127-134.

[16] H. Nguyen, and P. Haddawy, “The decision-theoretic video advisor,” In AAAI-98. Workshop on Recommender Systems, Madison, WI, 1998, pp. 77–80.