Recommender Systems for Intelligence Analysts

Anna L. Buczak[1], Benjamin Grooters[2], Paul Kogut[3], Eren Manavoglu[4], C. Lee Giles[4]

[1]Sarnoff Corporation, Princeton, NJ 08543
[2]Lockheed Martin Advanced Technology Laboratories, Cherry Hill, NJ 08053
[3]Lockheed Martin Integrated Systems & Solutions, King of Prussia, PA
[4]Pennsylvania State University, School of Information Sciences and Technology, University Park, PA
abuczak@sarnoff.com, bgrooter@atl.lmco.com, paul.a.kogut@lmco.com, manavogl@cse.psu.edu, giles@ist.psu.edu

Abstract

Homeland security intelligence analysts need help finding relevant information quickly in a rapidly increasing volume of incoming raw data. Many different AI techniques are needed to handle this deluge of data. This paper describes initial investigations in the application of recommender systems to this problem. It illustrates various recommender systems technologies and suggests scenarios for how recommender systems can be applied to support an analyst. Since unclassified data on the search behavior of analysts is hard to obtain we have built a proof-of-concept demo using analogous search behavior data in the computer science domain. The proof-of-concept collaborative recommender system that we developed is described.

1. Problem Description

Homeland security and other intelligence analysts spend too much time on the mechanics of retrieving relevant information and not enough time on deep analysis. Retrieval usually needs to be initiated by the analyst (i.e., information pull). Existing information push technologies (that automatically find information for the analyst) such as RDF Site Summary (RSS) have very course grained channels that lead to information overload. Retrieval becomes even more complex for analysts who require multi-INT data from diverse heterogeneous sources (e.g., text, imagery, geospatial). Intelligence analysts also spend too much time-sharing information in multi-organizational teams. Information sharing is usually accomplished by person-to-person interactions (e.g., phone calls, email). Collaboration tools like whiteboards and chat rooms still require significant human effort. We believe that recommender systems can help solve these knowledge management problems.

Assistant agents apply user profiles to proactively retrieve, share and recommend relevant information as shown in Figure 1. We are investigating hybrid aggregate knowledge representations and machine learning techniques for user profiles. Semantic Web based user profiles were described in [Kogut, 2004]. These Semantic Web based user profiles were entered by manually selecting classes and properties from a Web Ontology Language (OWL) ontology. This paper describes a recommender system approach that can automatically build a different form of user profiles, and can leverage existing profiles entered by the user. The approach takes advantage of similarities among user profiles for suggesting items of high importance to the analysts. The paper discusses ongoing efforts to develop and apply recommender systems for Homeland Security applications.

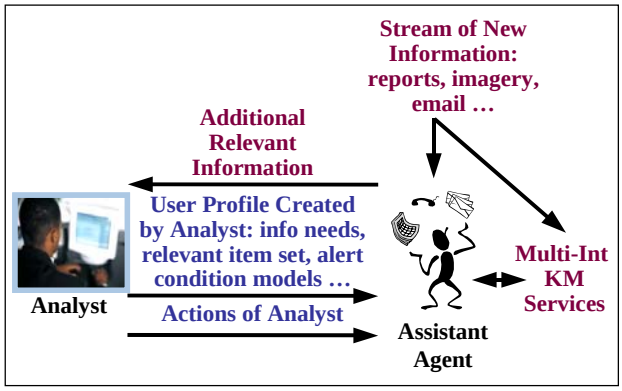

Figure 1. Assistant Agents for Knowledge Management

The paper is organized as follows. Section 2 describes different types of recommender systems; Section 3 talks about user profiles and recommender engines, and Section 4 describes adaptivity of recommender systems. Section 5 presents two-recommender system scenarios for intelligent analysts, and Section 6 describes the system that we developed. Section 7 discusses the conclusions.

2. Recommender Systems

Recommender systems are a form of artificial intelligence technology that provides the user with personalized suggestions about the items of interest to the individual, based on previous examples of the user's likes and dislikes. Recommender systems can suggest information of any type: web pages, news articles, books, movies, TV shows, images, news articles [Breese et al., 1998; Mooney, 2000; Buczak et al., 2002].

There are three main types of recommender systems:

- Collaborative Recommenders
- Content-Based Recommenders
- Hybrid Recommenders

Collaborative Recommenders, also called Collaborative Filtering Systems, make suggestions based on preferences of a set of users. They maintain the preferences of individual users usually in form of lists of items that the users liked or disliked. The Collaborative Recommender suggests documents that other users with similar profiles liked. In order to do that, it computes a measure of similarity between profiles of different users, and then suggests items that were of interest for other users with similar profiles. This recommender takes advantage of the experience of other users, suggesting items that other users liked. The advantages of a collaborative approach are that it performs well even with a small number of individual user ratings and it performs well in domains whose content is not easily analyzed by an automated process. The main disadvantage of this approach is that an item cannot be recommended unless at least one user has rated it.

A Content-Based Recommender makes suggestions based solely on the content of the items (without matching user's interests to interests of other users). In this type of recommenders each item is uniquely described by its content and each user is uniquely described by a user profile that describes the content that the individual likes. Content information can contain keywords and their frequencies, metadata related to a given item such as document author, title, place of publication, or actor (depending on the application). In Content-Based Recommenders usually a similarity measure is computed between a given item and the user profile. Items with the highest similarity are ranked and their list is presented to the user. The advantage of Content-Based Recommenders is that a new item can be recommended based solely on its content description. On the other hand a large number of previous ratings are necessary for the recommender to be accurate.

A hybrid recommender uses both the content-based and the collaborative recommendations, merging their results using one of the decision fusion methods (e.g., voting, weighted average, neural network). It combines the advantages of both content-based and collaborative recommenders. These include: a) Performing well even with a small number of individual user ratings; b) Performing well in domains whose content is not easily analyzed by the automated process (Collaborative part assures this); c) Recommendations for a new item can be done based on its content description (Content-based part assures this).

In a Homeland Security application a recommender system can reduce analyst's information overload, improve analyst's task success, decrease time to completion, expedite information dissemination, and foster collaboration between analysts. The recommender system for an intelligence analyst ought to suggest items useful to a given analyst in the task currently performed. Items of interest include web pages, news articles, other analysts' reports, queries to specialized databases, SIGINT data, HUMINT data, IMINT data, other imagery, etc.

For intelligence analysts, hybrid recommender systems are of interest since they take advantage both of the information content as well as the items that other analysts found useful in performing their tasks. This allows the analyst to take advantage of the methods (e.g., queries to specialized databases or to the internet) that other analysts developed, and the documents that other analysts with similar profiles found. The recommender system can foster additional collaboration between analysts by suggesting a given analyst to get in contact with other analysts whose profiles are similar and who work on a related task.

3. User Profiles and Recommender Engines

A recommender system is composed of two main parts: the user profiles and the recommender engine. A user profile describes the interests of a given user, while the recommender engine is the computational method that computes the predictions of how much interest a given item will be to a particular user. The computation performed by the recommender engine employs the user profiles stored in the system.

User profiles range from very simple to sophisticated. User profiles are usually simpler for collaborative recommenders, and contain more information for content-based recommenders. For hybrid recommenders they need to include both the collaborative and content-based parts. An example of a simple user profile is a list of items that the user found interesting (set of ids for documents in case of a collaborative recommender), possibly with the addition of how many times a given document was opened, saved, etc. It could also contain a list of items that were of no interest to the user. A more sophisticated user profile can contain some content information such as stems of the words used in user queries, or stems of the words employed in titles and abstracts of documents accessed by the user; again these could be augmented by relative frequency of the stems. Some other information that could be used in the user profile is the metadata describing the items of interest (such as genre in case of TV-shows or movies) [Buczak et al., 2002].

Up to now, we were assuming one profile per user, or at most one collaborative and one content-based profile per user. However in the intelligence analyst arena, one profile per analyst might not be sufficient given that analysts work on a variety of tasks and the type of documents of interest for each task can be very different. Alonso and Li [Alonso & Li, 2005] developed a system for intelligence analyst that builds a separate user profile (called context map) for each task that the analyst is performing. Structural elements of the context map include concepts and relations between them, both of which help capturing analyst's interests and concerns when performing a given type of task. Since for each type of task a different context map can be used, the system is very flexible. This flexibility is

increased by the adaptivity of the context maps to user feedback.

In an intelligent analyst application, a great value of a recommender system is to help the less experienced analysts performing their jobs. Since those new analysts do not have reliable profiles yet, as a way to bootstrap the system, a stereotypic profile could be used. A stereotypic profile for a certain task is a profile built from the profiles of the analysts that are experts at performing that task. A stereotypic profile can be obtained by some intelligent concatenation of expert profiles. Of course the exact method for doing it depends on the profile itself, what type of information it contains. For a profile that is a list of word stems and their frequencies, a stereotypic profile could be obtained by performing the union of all the word stems from individual profiles and then averaging the frequency of each of the stems across the profiles. The use of stereotypic profile for inexperienced analysts will allow them to receive recommendations about the items that would be of interest to a more experienced analyst.

Intelligence analysts often perform their tasks in a cooperative fashion meaning that not only individual profiles are of interest but also "cooperative" profiles that describe a team of analysts. How to build such a "cooperative" profile, what information it should contain, how different parts of the profile ought to work together, is a new area of research that needs to be addressed order to help analysts efficiently performing their work. It represents one of the challenges of using artificial intelligence for Homeland Security.

Based on user profiles and some description of the items to be eventually recommended, the recommender engine computes the predicted value of a given item to the user and arranges those predictions in an order meaningful to the user (e.g., highest predicted to lowest predicted value). There are a large number of recommender engine methods that can be employed. Some methods use neighborhood-based techniques that compute a measure of similarity between user profiles (weight) and subsequently use an equation to predict, based on this measure, how much a new item would be of interest to a given user. We are using this type of recommender engine in our application and it will be described in detail in Section 6. Other methods used for computing recommendations include Bayesian Classifiers [Billsus & Pazzani, 1998; Zimmerman et. al, 2004], Bayesian Networks [Breese et al., 1998], Decision Trees [Zimmerman et. al, 2004], Support Vector Machines [Zhang & Iyengar, 2002], case based reasoning [Balabanovic & Shoham, 1997; Cotter & Smyth, 2000], and induction rule learning [Basu et al., 1998].

4. System Adaptivity

One of the requirements for recommender systems is the adaptivity of the system i.e., the system learning the user preferences. The learning is performed based on the user feedback that can come as one of the two types: explicit and implicit. We deal with explicit feedback when the user explicitly states which items are and which are not of interest to him/her. The items of interest become positive examples, and the items that are of no interest become negative examples for the learning algorithm. Examples of explicit feedback are the Tivo™ "thumbs-up" and "thumbs-down" buttons – the more times a user presses the "thumbs-up" button, the more he/she likes the program. Explicit feedback is much easier to be used for adapting user profiles, since it has one meaning only (the user either found the item useful or not useful). Implicit feedback, deals with cases when the user does not give a rating to the item but the system needs to infer whether the user liked the item based on the actions that he/she took. In the analyst's world, the system can infer if a given document is of interest to the user based on the fact that is was opened, how long the analyst dwelt on it, the amount of scrolling performed, saving the document to the hard drive, etc. Since the amount of interest is inferred, there is a possibility of deducing the wrong information, making the task of learning from such feedback much more difficult.

System adaptivity can be achieved by adjusting user profiles themselves or by adapting the way the recommender engine works. Sometimes both methods are used in a recommender system. Most of the methods that adapt the user profiles perform it in such a way that when feedback about a certain item is given, this item (or some metadata about that item) with the corresponding "vote" is added to the profile. In case of a collaborative profile the id of the new, just rated document could be added, with a measure of how much the user liked it. The adaptivity of the system in this case is achieved by recalculating the measures of similarity between user profiles, subsequently leading to altered recommendations for the user.

In cases when adaptivity is achieved by adjusting the way the recommender engine works, the particular machine learning method used for this purpose depends on the recommender engine itself. For example if the recommender engine is an artificial neural network, this network is retrained using its particular training algorithm (e.g., error backpropagation in case of a multi-layer perceptron network) resulting in a new set of weights. Similarly for a recommender engine that is a Support Vector Machine (SVM), a kernel-based learning method is used that produces new support vectors, and thus a new SVM. In case of a Decision Tree recommender, a new decision tree is constructed usually using Quinlan's C4.3 algorithm. In those cases after learning a new neural network, a new decision tree, or a new SVM becomes the new recommender engine.

For certain applications, the recommender system needs to adjust a little bit, such as in case of a movie or TV-show recommendations. In those applications, user tastes and needs do not change significantly from week to week and the profile and the engine can be adjusted slowly. However in applications, such as intelligence analyst recommender systems, the adaptivity of user profiles and

recommender engines is the central requirement. Since the tasks on which the analyst is working can change rapidly, in order to be useful the system must learn user preferences quickly. At the same time, we do not want to start with an empty user profile every time the analyst starts a new task but we want to reuse as much information from the old profile as possible. These requirements mean that the system needs to learn preferences quickly from user feedback. Fast and reliable methods for learning analyst preferences represent the second challenge of using artificial intelligence for Homeland Security.

5. Recommender System Scenarios

In this section we will present two uses of a recommender system for intelligence analysts. The first scenario deals with recommender suggesting documents of interest to the analyst while the recommender in the second scenario suggests queries for the analyst to make to the system. This means that different information needs to be maintained in the user profiles in the two scenarios. We will depict this information when describing each of the scenarios.

The first scenario is graphically shown on Figure 2. The following steps are performed:

1. Our analyst, Major Brown queries the system about "Terrorism using backpacks". He is interested in relevant documents describing this type of terrorism.
2. Discovery Service (e.g., Google) performs the query on all the available databases and the internet.
3. Discovery Service (e.g., Google) passes results to the Recommender. The number of returned results is very large.
4. Recommender checks users with similar profiles, retrieves their documents that match the documents retrieved by the Discovery.
5. Recommender computes the recommendations for documents that match and rank orders them.
6. Recommendations above the threshold are presented to Major Brown. Recommender shows Major Brown the following list of recommendations:
   a. "Terrorism using bombs hidden in bags or luggage", author XX , 0.98
   b. "Suicide bombings using backpacks", author YY, 0.95
7. Major Brown has the option to click on any of the above documents or not to click. He clicks on "Terrorism using bombs hidden in bags or luggage".
8. The document "Terrorism using bombs hidden in bags or luggage" is displayed for Major Brown who views the data.

In the scenario described, Major Brown did not receive a list of a hundreds of items matching his query, but was able to receive just a short list of the most appropriate items, thanks to the recommender system. Those items were identified from all the items returned the by the first query,

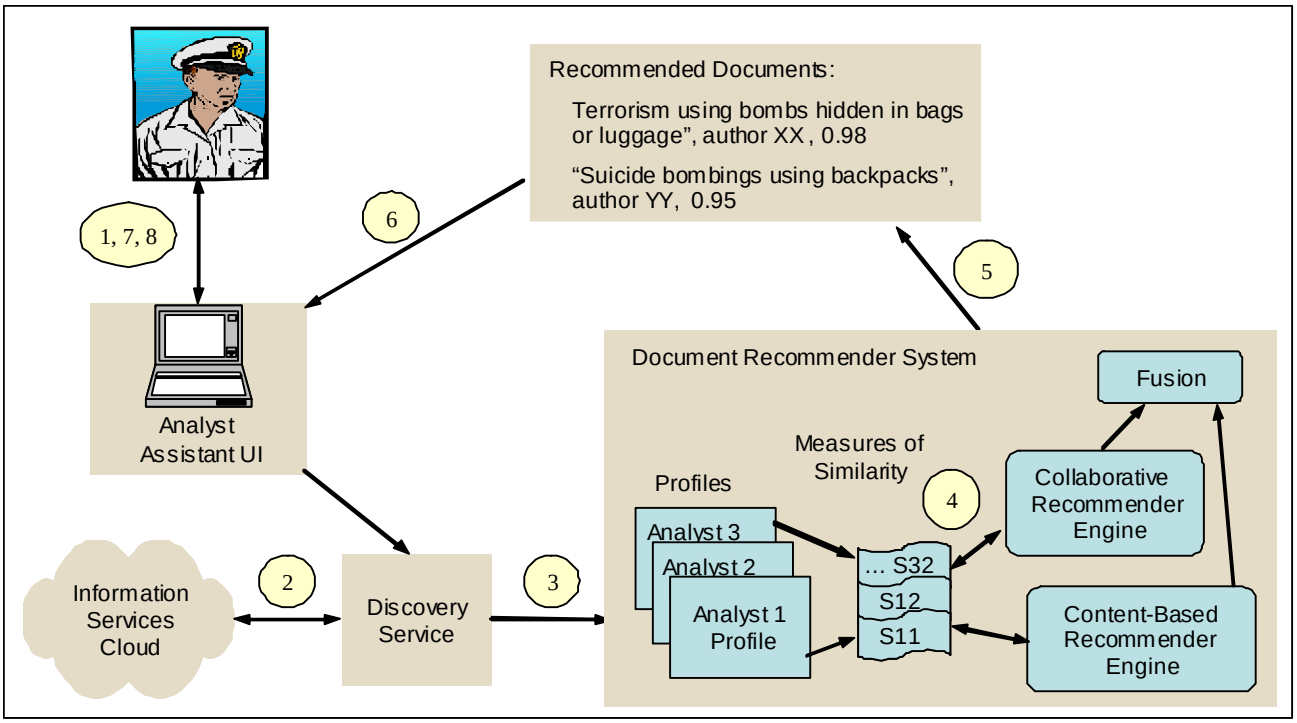

Figure 2. Hybrid Document Recommender System for Intelligence Analyst. Step numbers in the figure correspond to the numbers in Section 5 describing Scenario 1.

based on the items that other analysts with similar profiles found most useful.

The recommender used in Senario 1 is a hybrid system containing both a collaborative and content-based parts. The analyst profiles contain information about the documents that the analysts found of interest. The collaborative part of the profile contains ids of documents that the analyst found useful. Its content-based part encompasses metadata about the documents that were useful, word stems of the documents abstracts, titles, or of full document text.

In Scenario 2 the recommender is helping the analyst formulating a query. As such the information contained in the user profiles describes the queries that each of the analysts made previously. In lieu of document ids there are query ids, and instead of stems of words in the documents, there are stems of words in queries.

The second scenario is shown on Figure 3. The following steps are performed:

1. Our analyst, Major Black queries the system about "Terrorism using backpacks".
2. Discovery Service (e.g., Google) performs the query on all the available databases and the Internet and returns the results. Alternatively Seps 2-6 from Scenario 1 could be executed here, taking advantage of the Document Recommender System and thus providing the analyst with only a recommended subset of documents.
3. Results for the original query are returned to Major Black.
4. Major Black's original query is given to the Query Recommender.
5. Query Recommender checks analysts with similar profiles and retrieves their queries.
6. It computes the query suggestions for Major Black and rank orders them.
7. Recommendations above the threshold are presented to Major Black. System shows Major Black the following list of recommendations:
   - Ask about "Terrorism using Cars" 0.97
   - Ask about "Terrorism using Paint Cans" 0.92
8. Major Black has the option to click on any of the above queries or not to click. He clicks on "Terrorism using Cars".
9. A query on "Terrorism using Cars" is sent to Discovery and the results are passed to Major Black.
10. Major Black views the data.

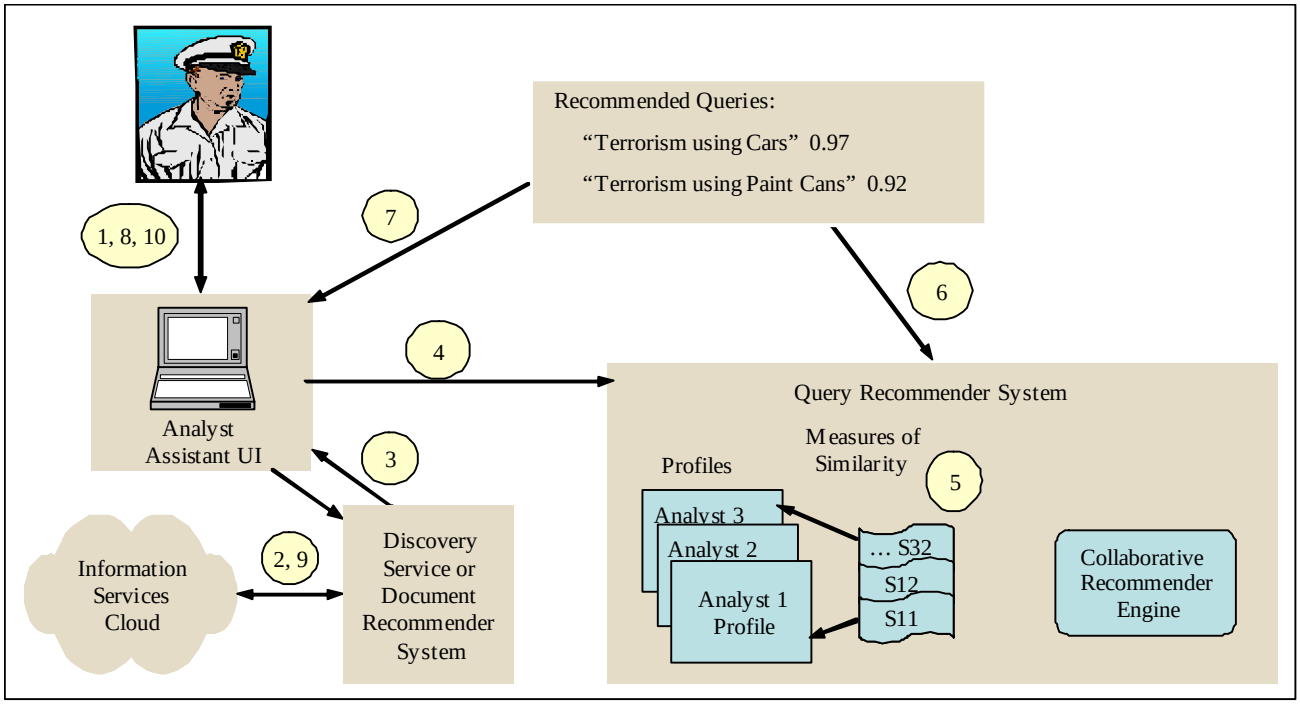

Figure 3. Collaborative Query Recommender System for Intelligence Analyst. Step numbers in the figure correspond to the numbers in Section 5 describing Scenario 2.

6. Proof of Concept Demonstration

To mimic the way we envision the recommender system working for the intelligence analyst (described in Section 5) we have developed a proof-of-concept recommender system working on anonyminized CiteSeer [citeseer] data. CiteSeer is an automated digital library and search engine of scholarly literature that provides access to the full-text of nearly 600,000 academic science papers, and over 10

million citations, primarily in the field of computer and information science. CiteSeer consists of three basic components: a focused crawler (harvester), the document archive and specialized index, and the query interface. The harvester crawls the web for relevant documents and after filtering crawled documents for academic documents; these are indexed using autonomous citation. Automatic extraction of the context of citations allows researchers to determine the contributions of a given research article quickly and easily; and several advanced methods are employed to locate related research based on citations, text, and usage information. CiteSeer is a full text search engine with an interface that permits search by document or by numbers of citations.

CiteSeer receives over a million hits a day and stores this hit information for research purposes. This anonyminized data is exactly the data that we used for developing the recommender system. The data set that we received from Penn State contained anonyminized CiteSeer data logs spanning nine months (July 2001- March 2002) and describing user interactions with the system. We developed software to split the original log file into separate files for each user (about 154,000 different anonyminized users); subsequently we split those files into individual "sessions" defined as sets of activities with less than an hour in between each activity. We further filtered the data by removing users who had less than 10 separate sessions and users who did not perform any queries about titles or authors. On the remaining data we performed a user pair-wise comparison and choose a subset of users who had at least 10 documents in common with the remaining users in the set. This ensured that the subset chosen was not sparse anymore and that it could be used for developing a collaborative recommender system. The subset of users chosen in this fashion for our experiments included 99 users.

The collaborative user profiles that we developed contained the document ids of all the documents that the user displayed or saved, the number of times the user displayed, and the number of times the user saved a given document. The implicit vote that we assigned in the system was 1.0 if the user saved the document, 0.8 if he/she displayed the document but did not save it, and 0.0 otherwise.

For each user we computed a measure of similarity with each other user using Average of Pearson Correlation and Extended Pearson (APEP) measure of similarity that we defined as:

$$\mathrm{APEP}_{a,i} = \frac{\mathrm{Pearson}_{a,i} + \mathrm{ExtendedP\ earson}_{a,i}}{2} \quad (1)$$

where $\mathrm{Pearson}_{a,i}$ is the correlation measure [Resnick et al., 1994] between the active user *a* and the user *i*:

$$Pearson_{a,i} = \sum_{j} \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}} \quad (2)$$

where $v_{i,j}$ is the vote of user *i* on item *j*, I_i denotes the set of items for which user *i* voted, and summation *j* goes over all the items for which both users *a* and *i* have recorded votes. The Pearson correlation takes into account only the items for which both users *a* and *i* have recorded votes (i.e., intersection of I_a and I_i).

$\mathrm{ExtendedPearson}_{a,i}$ takes into account the default voting extension proposed in [Breese et al., 1998]. This extension was introduced based on the observation that for sparse data sets, when users share only a few items of interest in their profiles, Pearson correlation is not reliable because it works on the intersection of the items that the two individuals voted on (I_a and I_i). The default voting extends the correlation measure to the union of I_a and I_i. All the unknown votes that were added (i.e., votes for user *a* for documents that were in the profile of user *i*, and votes for user *i* for documents that were in the profile of user *a*) had a vote set to 0.4. 0.4 reflects in our application an unknown vote since we use a vote of 0.8 for documents that were displayed but not saved by the user. When computing the Extended Pearson we additionally take into account that there is a number of additional items (*k*) on which neither user has votes but on which nonetheless they would agree upon [Breese et al., 1998]. For those *k* items we add 0.4 votes for both users. With all the extensions described above the equation for computing $\mathrm{ExtendedPearson}_{a,i}$ becomes:

$$ExtendedPearson_{a,i} = \frac{(n+k)(\sum_j v_{a,j} v_{i,j} + 0.16k) - (\sum_j v_{a,j} + 0.4k)(\sum_j v_{i,j} + 0.4k)}{\sqrt{(n+k)(\sum_j v_{a,j}^2 + 0.16k) - (\sum_j a_{a,j} + 0.4k)^2 ((n+k)(\sum_j v_{i,j}^2 + 0.16k) - (\sum_j v_{i,j} + 0.4k)^2)}} \quad (3)$$

where all the summations are taken over the union of items that either user *a* or *i* has voted on; *n* is the number of items in this union, and *k* is number of documents in the intersection of I_a and I_i.

We compute the recommendation for the active user *a* for item *j* ($predicted_{a,j}$) as:

$$predicted_{a,j} = \bar{v}_{a,j} + \kappa \sum_{i=1}^{m} APEP_{a,i} (v_{i,j} - \bar{v}_i) \quad (4)$$

where $\bar{v}_i$ is the average of user *i* votes, κ is the normalizing factor, $\mathrm{APEP}_{a,i}$ is the similarity of active user to user *i* (computed using eq. 1), $v_{i,j}$ is the vote of user *i* on item *j*, and *m* is the number of users in the set with nonzero $APEP_{a,i}$ weights.

The scenario in which we used the developed recommender system was very similar to the intelligent analyst scenario presented in Figure 2 and described in Section 5. The only difference between those scenarios

was that all calls by "discovery service" to the internet were substituted by CiteSeer API calls to CiteSeer. Also since at this point we have only the collaborative recommender implemented, this recommender was used instead of the hybrid recommender presented on Figure 2. We are in the process of testing the performance of the developed recommender system and it will be described in a separate publication.

7. Conclusions

Recommender systems are an important artificial intelligence technology for helping intelligence analysts deal with information overload. Recommenders can filter out much of the data that is not relevant for an analyst when performing a given task and thus make the work of the analyst easier. Collaborative and hybrid recommenders allow analysts to automatically take advantage of the knowledge and experience of other analysts, making them especially interesting for use by novice analysts. In order for recommender systems to be useful in a Homeland Security application they cannot be static but the analysts' profiles and the recommender engines need to be dynamically updated using machine-learning techniques. The analyst should be able to give feedback to the system, telling it that certain suggestions were good or bad. This feedback should be used by machine learning techniques for updating the profiles and improving the recommendations over time.

Two particular challenges of using artificial intelligence for Homeland Security that we identified in the area of recommender systems are "cooperative" user profiles and fast machine learning methods for adapting analyst preferences. What information "cooperative" user profiles should contain, how different parts of the profiles should work together are just a few of the important research areas identified. Machine learning methods for adapting user profiles and recommender engines in a fast, reliable, and preferably incremental fashion are important research areas to be addressed as well. The synergy of recommender systems technology with the Semantic Web should be also investigated.

5. References

//citeseer.ist.psu.edu/

Alonso, R., H. Li, (2005), "Combating Cognitive Biases in Information Retrieval", submitted to *First International Conference on Intelligence Analysis Methods and Tools*, McLean, Virginia.

Balabanovic, M., Y. Shoham, (1997), "FAB: Content-Based Collaborative Recommender", *Communications of the ACM*, 40(3), pp. 62-72.

Basu, C., H. Hirsh, W. Cohen, (1998), "Recommendations as Classification: Using Social and Content-Based Information in Recommendation", *Workshop on Recommender Systems*, AAAI Press, pp. 11-15.

Billsus, D., M.J. Pazzani, (1998), "Learning Collaborative Information Filters", *Fifteenth International Conference on Machine Learning*, Wisconsin, USA, pp.46-54.

Breese, J.S., D. Herlocker, C.Kadie, (1998), "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", *Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI, Morgan Kaufman.

Buczak, A.L., J. Zimmerman, K. Kurapati, (2002), "Personalization: Improving Ease-of-Use, Trust and Accuracy of a TV Show Recommender", *International Conference on Adaptive Hypermedia and Adaptive Web Based Systems*, *2nd Workshop on Personalization in Future TV*.

Cotter, P., B. Smyth, (2000), "PTV: Intelligent, Personalized TV Guides", *Seventeenth National Conference on Artificial Intelligence*, Austin, TX, USA, pp. 957-964.

Kogut, P., J. Yen, Y. Leung, S. Sun, R. Wang, T. Mielczarek, B. Hellar, (2004), "Proactive Information Gathering for Homeland Security Teams," *Communications of the ACM*, Vol. 47, No. 3, pp. 48-50.

Mooney, R. J., Roy, L., "Content-Based Book Recommending Using Learning for Text Categorization", *Fifth ACM Conference on Digital Libraries*, pp. 195-240, San Antonio, TX, June 2000.

Resnick, P., N. Ivacovou, M. Suchak, P. Bergstorm, J. Riedl, (1994), "Grouplens: An Open Architecture for Collaborative Filtering of netnews", *ACM 1994 Conference on Computer supported Cooperative Work*, pp. 175-186, New York, ACM.

Zhang, T., V.S. Iyengar, (2002), "Recommender Systems Using Linear Classifiers", *Journal of Machine Learning Research*, 2(2002), pp. 313-334.

Zimmerman, J., K. Kurapati, A.L. Buczak, D. Schaffer, J. Martino, S. Gutta, (2004), "TV Personalization System: Design of a TV Show Recommender Engine and Interface", eds. L. Ardissono, A. Kobsa, M. Maybury, *Personalized Digital Television: Targeting Programs to Individual Viewers*, pp. 27-52, Kluwer: the Netherlands.