A Recommender System Based on the Collaborative Behavior of Bird Flocks

Esin Saka
Knowledge Discovery and Web Mining Lab,
University of Louisville
Louisville, KY, USA
Email: esin.saka@louisville.edu

Olfa Nasraoui
Knowledge Discovery and Web Mining Lab,
University of Louisville
Louisville, KY, USA
Email: olfa.nasraoui@louisville.edu

***Abstract*—This paper proposes a swarm intelligence based recommender system (FlockRecom) based on the collaborative behavior of bird flocks for generating Top-N recommendations. The flock-based recommender algorithm (FlockRecom) iteratively adjusts the position and speed of dynamic flocks of agents on a visualization panel. By using the neighboring agents on the visualization panel, top-n recommendations are generated. The performance of FlockRecom is evaluated using the Jester Dataset-2 [1] and is compared with a traditional collaborative filtering based recommender system. Experiments on real data illustrate the workings of the recommender system and its advantages over its CF baseline.**

***Index Terms*—Swarm intelligence; recommender system; collaborative filtering; flocks of agents; bird flocks.**

I. Introduction

In recent years, we have witnessed an explosive growth in the amount of information. Each day, more books and journals are published, more newspaper articles are written, more web pages are posted online, more office documents are prepared, more photos are taken, and more movies are created.

This over-abundance of information contributes to the reasons why we can get hundreds or even thousands of results for a simple search, and why we can find it hard to arrive at the resources that we need by wading through endless labyrinths of Web pages and Websites. This problem is commonly referred to as *information overload*.

Recommender systems aim to assist users in handling the information overload problem. Two of the many approaches to build recommender systems include collaborative filtering (CF) and swarm intelligence (SI), both built on the collaboration of users. Based on the assumption that users with similar past behaviors (rating, browsing, or purchase history) have similar interests, a collaborative filtering system recommends items that are liked by other users with similar interests [2], [3], [11]. More information on CF and other approaches are presented in Section II. In this paper we present a new recommender system approach using a swarm intelligence algorithm, inspired from bird flocks and called flocks-of-agents based recommender system (FlockRecom). In this approach, each user is mapped to one agent, i.e. each agent of the flock represents a user. Initially, agents are placed on a planar surface (hereinafter referred to as the *visualization panel*). Then, in each iteration, similar agents attract each other, while dissimilar agents repel each other. Thus, the agents' speed gets updated according to their neighboring agents. In time, similar agents start moving together and closer, forming clusters [4]–[6]. Moreover, the distance between the agents depends on the similarity between the users that are mapped to those agents. At each iteration, recommendations are generated/updated using the neighboring agents. Agents keep moving until they are forced to stop. Thus, the dynamic character of the FlockRecom provides dynamic recommendations, making FlockRecom stronger at exploring different recommendation options and providing more variety for recommendations. Variety or diversity is important in the environments that users keep visiting repeatedly. For example, on Facebook[1], neither giving the same suggestions over and over, nor random suggestions may satisfy the users. Additionally, initial experimental results show that, FlockRecom is a promising approach for recommendation in dynamic environments, and in the future, in social networking platforms.

In this research, we start by reviewing recommender systems in Section II, then bird flocks in Section III. We then present FlockRecom in Section IV. In Section V, we present experiments on a real life dataset. Finally, we make our conclusions and discuss future work in Section VI.

II. Recommender Systems Overview

The Human brain is a fast and intelligent decision making organism, but when significant information overload is encountered, some additional external guidance may be needed. Systems that strive to achieve this aim may be known under different names such as decision support systems, recommender systems, customer relationship management systems, executive support systems, executive information systems, or personalized agents. Figure 1 shows the basic modules of a typical recommender system.

A recommender system can analyze the data to compute recommendations in different ways, including:

1) *Content-based or Item-based filtering*

This work is supported by National Science Foundation CAREER Award IIS-0133948 to Olfa Nasraoui.

[1] http://www.facebook.com/

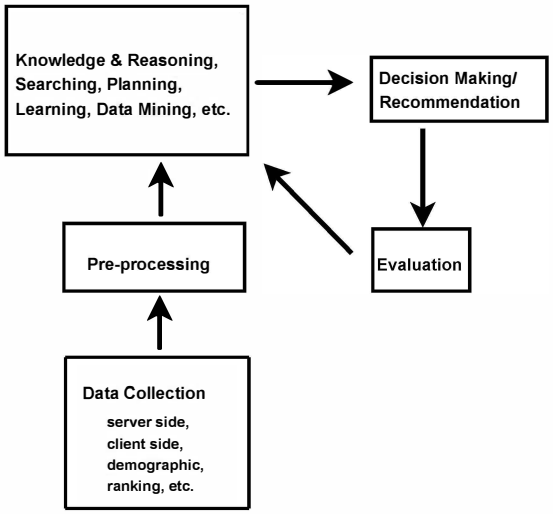

Fig. 1. Modules and flow of typical recommendation systems.

2) *Collaborative filtering*
3) *Knowledge Engineering or Rule-based filtering*
4) *Demographic filtering*
5) *Hybrids*

Content-based filtering systems recommend items to a given user, which are deemed to be similar to the items that the same user liked in the past or similar to the user profile in attributes. Item similarity is typically based on domain specific item attributes (such as author and subject for book items, artist and genre for music items), which are thus part of the items or user profile. Classical examples include Syskill and Webert [7], and Fab [8]. This approach has the advantage of easily including brand new items in the recommendation process, since there is no need for any previous implicit or explicit group user ratings or purchase data to make recommendations. Content-based filtering systems suffer from several limitations. First, they tend to be limited to certain types of content such as text and movies. Even in this case, the extracted features are limited in that they only capture certain aspects of the content. Second, they tend to provide over-specialized recommendations based only on user profiles or items that are similar to items previously rated by the user. Hence, users cannot explore new items that are different from those included in their profiles.

Based on the assumption that users with similar past behaviors (rating, browsing, or purchase history) have similar interests, a collaborative filtering system recommends items that are liked by other users with similar interests [2], [3]. This approach relies on a historic record of all user interests such as can be inferred from their ratings of the items on a website (products or web pages). Rating can be explicit (explicit ratings, previous purchases, customer satisfaction questionnaires) or implicit (browsing activity on a website or clickstreams). Typical examples include GroupLens [9] [10] and a survey can be found in [2]. The recent Netflix price increased the attention on collaborative filtering[2] [11]. Collaborative filtering can be either user-based or item-based. In user-based collaborative filtering, historic data such as purchases, visits, or ratings of items such as products or web pages, is used to form user neighborhoods of similar users. Later, for a new user, items are recommended if they are liked by this user's neighbors. In item-based collaborative filtering, historic data is used to form associations between items that tend to be liked by the same user. Later, for a new user with known ratings for a few items, other items that are associated with the known rated items are recommended.

In knowledge engineering or rule-based filtering approach, used frequently to customize products on e-commerce sites, the user answers several questions, until receiving a customized result such as a list of products or a custom-built configuration of a product (e.g. Dell's web page[3]). This approach is mostly based on heavy planning of a judicious set of questions and possible answer combinations by an expert, and establishing this dialog depends on manually coded scenarios that assume heavy knowledge about how each item fills the needs of a particular user.

In demographic recommender systems approach, items are recommended to users based on their demographic attributes, such as gender, age, location, salary, etc. The recommendations can be based on handcrafted stereotypes derived from marketing research or on machine learning techniques [12] that learn to predict users' preferences from their demographic attributes. For instance, users can be classified into one of several classes based on their personal attributes, and this class information can form the basis for recommendations.

III. Bird-Flocks in Computing

The inspiration behind the flocks-of-agents-based recommender system stems directly from the collaboration among bird flocks in nature. One of the definitions given for a *flock* is "*a number of animals of one kind, esp. sheep, goats, or birds, that keep or feed together or are herded together*"[4].

"The motion of a flock of birds is one of nature's delights" according to Craig Reynolds who has first simulated this phenomenon in computer animation, where the bird-like, birdoid object was called *boid* [13]. One of the biggest differences between a particle and a boid in simulation is that boids have *orientation*, which makes them suitable for *data visualization* as well as clustering and recommender systems.

Studies about flocks of agents in computer science have mainly started with simulating moving bird flocks, based on two balanced and opposing behaviors of natural flocks, namely, 1) Desire to stay close to the flock, and 2) Desire to avoid collisions. These are simulated in the following three behaviors [13].

Natural Bird Flock Behaviors:

1) **Collision Avoidance/Separation:** Steering away from the other boids to avoid collision.

[2]http://www.netflixprize.com
[3]http://www.dell.com/
[4]http://dictionary.reference.com/browse/flock

2) **Alignment/Velocity Matching:** Aiming to match the moving direction (i.e. heading) and speed to that of nearby flockmates.
3) **Cohesion/Flock Centering:** Attempting to adjust steering toward the average position of local flockmates and to stay close to the neighbors.

While cohesion and velocity matching represent the attraction forces, which keep the boids together, collision avoidance formed the rejection/repelling force. Other studies also tried to present behavioral rules and model collective behavior of animals [14], [15]. Later studies also focused on visualizing data using flocks of agents. Each individual boid represented one data item and a fourth behavior was added to represent moving with similar data items [16]:

4) **Information Flocking:** Attempting to move with similar boids.

The fourth behavior is pretty similar to the second behavior, velocity matching. However, in the fourth behavior, the aim is not moving together with all neighbors, but only with the ones similar enough to form a group. This behavior provided a suitable ground for using flocks of agents for data visualization and offered a motivation for data clustering [4]–[6]. In this research, it is the information flocking which powers the recommender system.

It should be noted that, just as a flock can be formed of birds, it can also be formed by other boids such as fish, sheep, etc. Therefore, for the sake of generality, in this study, instead of the word boid, we use the word "agent".

IV. Flocks-of-Agents based Recommender System

Flocks of agents-based recommender system is suitable for any kind of data set where one can define a similarity measure between users. In the flocks of agents based recommender system approach, each agent represents one user. Initially, agents are placed on a *visualization panel*, which is a 2 or 3-dimensional continuous space, where x, y (and if applicable z) coordinate values range between 0 and 1. Agents may be placed randomly or some background information can be used to place them. Then, they start moving around. As they meet other agents in a defined neighborhood, they try to remain at an ideal distance to each other, which is determined according to the similarity of the users that agents are representing. The more the users are similar, the smaller the ideal distance will be. Ideal distances are computed for each agent pair once at the beginning of the algorithm based on the intrinsic properties or ratings of the users. If neighboring agents are further apart than the ideal distance, there will be an attraction force between them and the agents will try to move closer to each other. In contrast, if the distance is less than the ideal distance, then there will be a rejection force, and agents will move apart from each other. Given this basic idea, Algorithm 1 gives the procedure for *Flocks of Agents Recommender System (FlockRecom)*.

In steps 1 and 2, the initialization is performed. The velocity vector $\overline{v}$, is a unit vector, (i.e. $||\overline{v}|| = 1$), representing the direction. In step 3, the ideal distances between agents are computed via Equation (1), where $sim(i,j)$ is the similarity between the users that agents i and j are representing, and sim_{th} is the similarity threshold.

$$d_{ideal}(i,j) = \begin{cases} \frac{1-sim(i,j)}{1-sim_{th}} \times d_{th}, & sim_{th} \neq 1 \\ 0, & sim_{th} = 1 \end{cases} \quad (1)$$

Later, for each agent i, the neighboring agents that are close enough to i on the visualization panel, are extracted in Line 6, where *d(i,j)* is the 2D Euclidean distance between agents i and j, and d_{th} is the distance threshold. Then, for each neighbor:

- If the distance between the agents i and j is equal to the ideal distance between them (Line 7), there is no attempt to change i's velocity due to j (Line 8).
- If the distance between the agents i and j is greater than the ideal distance between them (Line 9), an attraction force will move i closer to j, with a more similar velocity to j (Line 10).
- If the distance between the agents i and j is smaller than the ideal distance between them (Line 11), a repelling force will move i further from j, with a less similar velocity to j (Line 12).

In line 14, the velocity effect on i due to neighbor j is computed where $\overline{v}_{cap}(i,j)$ is the unit vector pointing from i to j. Next is the computation of the updated velocity of agent i, $\overline{v}_{next}(i)$, between lines 16 and 25. First, if i has neighbors, then their resulting velocities on i are summed up and normalized. If the total, normalized velocity $\overline{w}$, does not change the agent's current direction more than 90 degrees, then the updated velocity is assigned as $\overline{w}$. Otherwise the velocity is kept unchanged for the next iteration. Similarly, if agent i does not have any neighbors -note that an agent is not considered to be a neighbor of itself- then the velocity will be kept the same for the next iteration. In line 26, the amplitude is computed depending on the number of neighbors and distance threshold, where amp_{def} is the default minimum amplitude. The minimum amplitude is empirically set to 0.1. At the end of each iteration, the updated agent coordinates are computed, and all the agents are moved to their updated positions simultaneously (lines 28 to 31). Then the next step is to generate recommendations. Let's call *active users* to the users that recommendations are generated for. For each active user, a set of users are determined via the agent neighborhood on the visualization screen. In other words, for the active user u's corresponding agent i_u, neighbor agents set $S(j)$ is determined such that $d(i,j) \leq d_{th}$ and $i \neq j$. Then, average item ratings are computed for all users represented by the agents in $S(j)$. Finally, n items with the highest average ratings are recommended to user u. These n items are called *top-n* items.

V. Experiments

The experiments were conducted on a dataset extracted from the Jester Dataset-2, which is available online on the website of the University of California, Berkeley [1]. In

Algorithm 1 FlockRecom Algorithm

Input: Dataset.
Output: Top-N recommendations.

Initially place the agents on the visualization panel
Initialize velocities of all agents
Compute the ideal distances, d_{ideal}, between agents.
while 1 **do**
  for each agent i **do**
    for all j such that $d(j,i) \leq d_{th}$ and $i \neq j$ **do**
      if $d(i,j) = d_{ideal}(i,j)$ **then**
        $\beta(i,j) \leftarrow 0$
      else if $d(i,j) > d_{ideal}(i,j)$ **then** {attraction}
        $\beta(i,j) \leftarrow 4 \times \left(\frac{d(i,j) - d_{ideal}(i,j)}{d_{th} - d_{ideal}(i,j)}\right)^2$
      else {repulsion}
        $\beta(i,j) \leftarrow -4 \times \left(1 - \frac{d(i,j))}{d_{ideal}(i,j)}\right)^2$
      end if
      $\overline{v}_{resulting}(i,j) \leftarrow \overline{v}(j) + \beta(i,j) \times \overline{v}_{cap}(i,j)$
    end for
    if $\exists\ j$ such that $d(j,i) \leq d_{th}$ and $i \neq j$ **then**
      $\overline{w}(i)$ = normalize ($\sum_{j|d(j,i) \leq d_{th} \& i \neq j} \overline{v}_{resulting}(i,j)$)
      if The angle between $\overline{v}(i)$ and $\overline{w}(i)$ is less than or equal to 90 degrees **then**
        $\overline{v}_{next}(i) \leftarrow \overline{w}(i)$
      else
        $\overline{v}_{next}(i) \leftarrow \overline{v}(i)$
      end if
    else
      $\overline{v}_{next}(i) \leftarrow \overline{v}(i)$
    end if
    amp_{next}(i) $\leftarrow amp_{def} + \frac{\left|\sum_{j|d(j,i) \leq d_{th} \& i \neq j} \overline{v}_{resulting}(i,j)\right|}{100}$
  end for
  for each agent i **do**
    compute new position $p_{next}(i) \leftarrow p_{current}(i) + amp_{next}(i) \times \overline{v}_{next}(i)$
  end for
  Move all agents to the updated positions and update current velocities.
  for Each user u that will be provided recommendation **do**
    for The agent i_u, that represents u **do**
      for Users represented by neighboring agents' of i_u **do**
        Compute the average ratings per item
        Recommend Top-N items to u
      end for
    end for
  end for
end while

addition to FlockRecom, a traditional collaborative filtering based recommender system was also implemented to test the performance of our system, and performances were evaluated and compared using precision, recall, and F1 metrics.

A. Dataset

In our experiments, we used the Jester Dataset-2, which is a collection of user ratings for 150 different jokes [1]. The dataset has 63,978 users, and the ratings range on a real value scale from -10 to +10 (-10 and +10 are included). As shown

TABLE I
DATASET.

Dataset ID	Number of Users	Number of Items	Avg. Sim.
Jester	50	150 (Jokes)	0.29

in Table I, in the experiments, the first 50 users were used with all 150 jokes, thus 1911 ratings were used.

B. Pre-Processing

The user ratings for jokes were in the scale of -10 to +10. In the pre-processing phase these ratings were normalized in the scale 0 to 1, where 0 indicates that the item is rated -10 or is not rated by the corresponding user.

C. Evaluation Metrics

Evaluating a recommender system can be nearly as hard as designing and implementing the system, in part because no simple, objective, and general agreed upon mathematical formula is always available to measure success [2], [17]–[20]. One problem suffered by some systems is *over-specialization*. When the recommendations are limited to the user's behavior, user's profile, or user's ratings, the user can be restricted to seeing only similar items, and there will be no randomness. In artificial intelligence, this problem is known as, the exploration/exploitation dilemma.

Evaluating information retrieval systems can be done if one has available, a set of user queries and a labeled set of search results (relevant and non-relevant). In this case, *precision* (proportion of retrieved items that are really relevant) and *recall/coverage* (proportion of all items, known to be relevant, that are retrieved) are typically used as goodness metrics [17], [18]. One method for evaluating a recommender system is asking for a ranking or a rating of the results from the users. However, this can be subjective. Moreover, if the study is for research purposes, it can be hard to find a sufficient number of real users with diverse interests for the experiment. For this reason, historical data has also been used in research studies. In this case, the output of a recommender system is compared to the real moves of the user in the historical data, and metrics such as precision and coverage are computed [17], [18]. One popular way to assess the success of a system is to compare it, for example with recommending the default, most popular, or even a randomly selected item.

As a summary, recommendations should be as close as possible to the real moves of the user. Closeness should take into account both (i) precision (a recommendation list's items are all correct or included in the original input data, i.e. they include only the true data items) and (ii) coverage/recall (a recommendation list's items are complete compared to the user moves, i.e. they include all the data items). The equations for precision and recall are given in Equation 2 and Equation 3, respectively.

$$precision = \frac{\text{size of suggested and relevant items}}{\text{size of all relevant items}} \tag{2}$$

$$recall = \frac{\text{size of suggested and relevant items}}{\text{size of all suggested items}} \tag{3}$$

A precision score of 1 indicates that every recommendation retrieved was relevant, whereas, a recall score of 1 represents that all relevant recommendations were retrieved. Since both of these contradicting measures are important in a recommender system, they can be combined in the F1 measure, given in Equation (4). Higher values of the F1 measure indicate a more balanced combination of high precision and recall.

$$F1 = \frac{precision \times recall}{precision + recall} \times 2 \tag{4}$$

D. Experimental Results

In the experiments, using FlockRecom, we generated top-n recommendations for active users at each iteration of our dynamic FlockRecom algorithm, n ranging from 1 to 30.

To compare the results, a traditional user-based nearest neighbor algorithm [2] was used. In the collaborative filtering approach, the neighbors of an active user u are defined as the users that are similar to u above a similarity threshold sim_{th} as given in Equation 5. After the neighbors are computed, the average item ratings per active user are evaluated using Equation 6, where u is the active user and i is an item.

$$CF\_neighbor(u) = \{t | similarity(u,t) \geq sim_{th}, t \neq u\} \tag{5}$$

$$Average\_rating(u,i) = \frac{\sum_{t \in \text{CF\_neighbor(u)}} rating(t,i)}{\text{number of neighbors}} \tag{6}$$

In the experiments, the cosine similarity was used to compute the similarity between users and the similarity threshold, sim_{th}, was set to 0.07, whereas the distance threshold d_{th} was set to 0.4, empirically.

The evaluation metrics were averaged over 10 different active users and 10 different runs per active user.

Figures 2 to 4 show the results of evaluations for the flocks-of-agents based recommender system (FlockRecom - FR) in comparison to the results of collaborative filtering (CF). The figures display the quality versus iteration or time.

Comparing FR to CF in Figure 2, we observe that the precision values for FlockRecom are slightly better than those for CF, especially, for small N. Additionally, FlockRecom provides more variety in the recommendations. Similarly, as seen in Figure 3, FlockRecom produced slightly higher recall values than CF. As expected, both for FlockRecom and CF, recall values increased as N, the number of recommended items, was increased. As a result, the F1 metric was higher for FlockRecom, especially for $N = 5$, as Figure 4 shows.

The fluctuations in Figures 2 to 4 are due to the exploration in FlockRecom, thus showing that, unlike CF, FlockRecom does not recommended the same items over and over. Figure 5 presents the number of times each joke is recommended

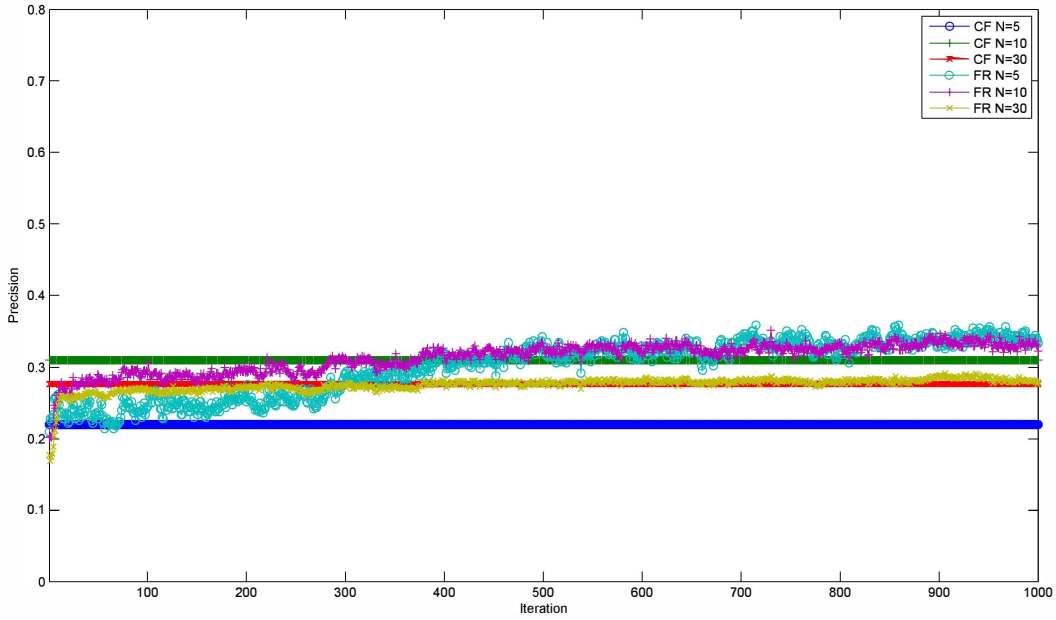

Fig. 2. Flock-based recommender system (FR) compared to standard collaborative filtering-based recommender system (CF) based on the average precision values for different numbers of top-n recommendations, over time. Averaged over 10 different active users, 10 different runs per active user. The x-axis represents the iteration number.

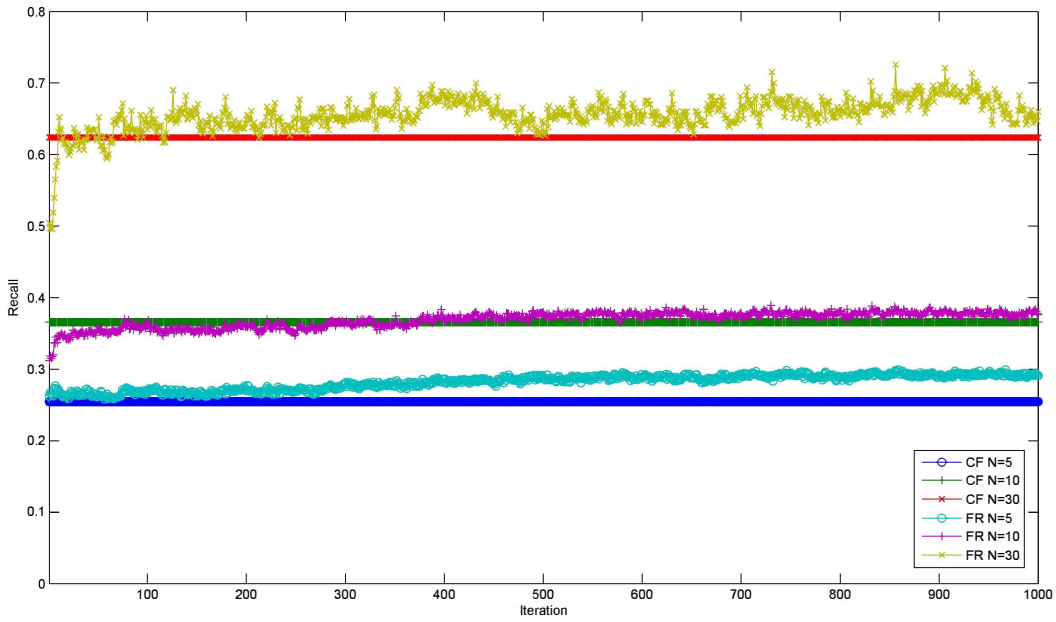

Fig. 3. Flock-based recommender system (FR) compared to standard collaborative filtering-based recommender system (CF) based on the average recall values for different numbers of top-n recommendations, over time. Averaged over 10 different active users, 10 different runs per active user. The x-axis represents the iteration number.

for a specific user over 10 different runs. While CF kept recommending the same items, FlockRecom added exploration and variety without losing quality.

To sum up, FlockRecom produced slightly better results than CF, after a sufficient number of iterations. For small values of N (which is preferred to avoid overloading users

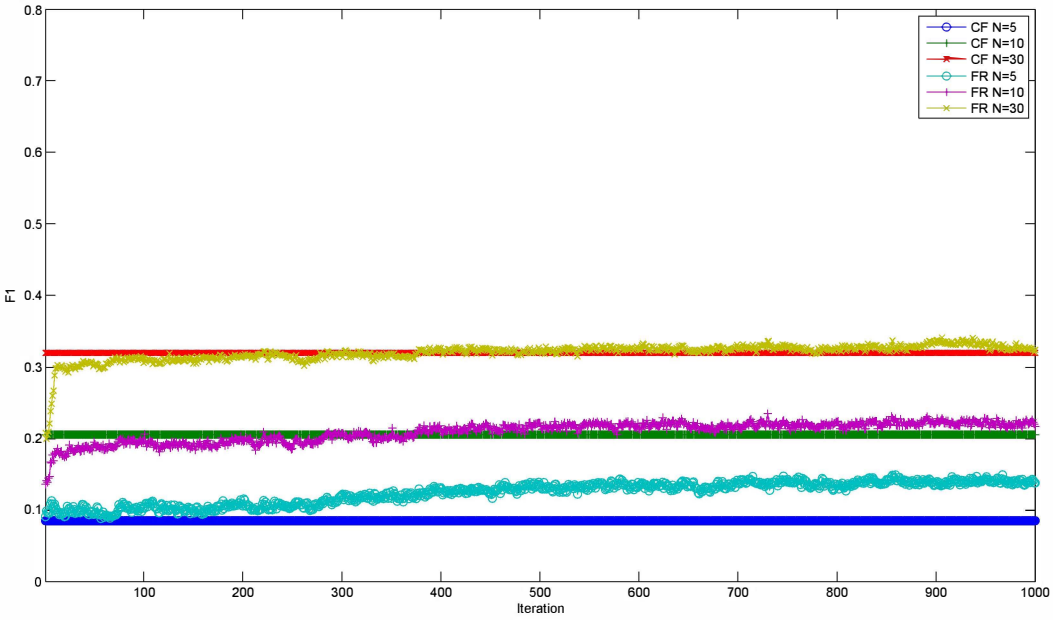

Fig. 4. Flock-based recommender system (FR) compared to standard collaborative filtering-based recommender system (CF) based on the average F1 values for different numbers of top-n recommendations, over time. Averaged over 10 different active users, 10 different runs per active user. The x-axis represents the iteration number.

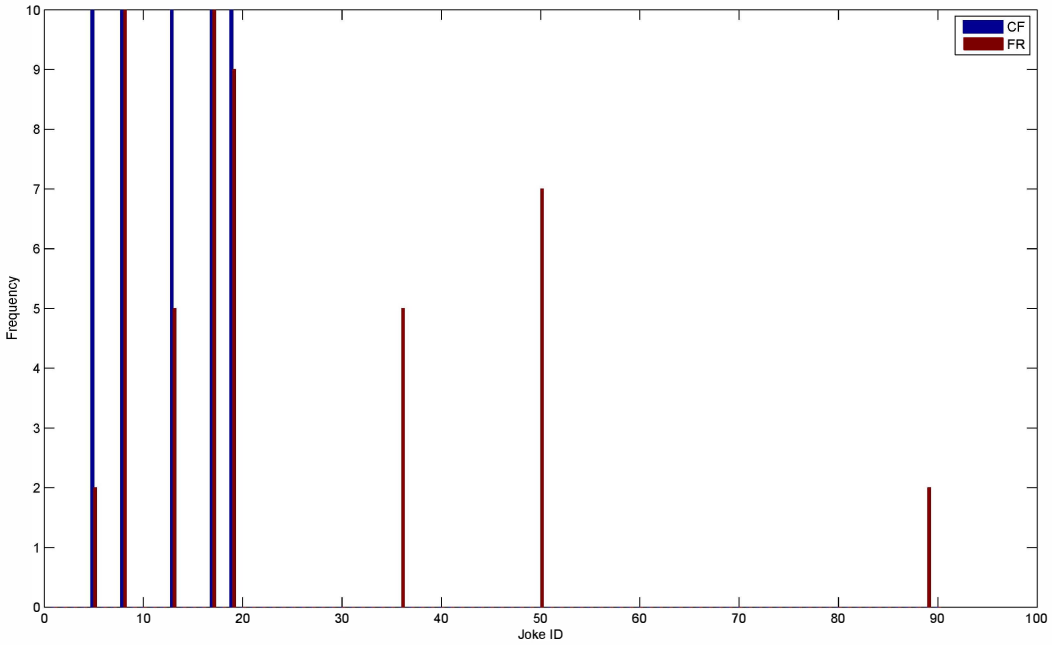

Fig. 5. The variety in the recommended items of Flock-based recommender system (FR) compared to standard collaborative filtering-based recommender system (CF) for different numbers of top-5 recommendations, at iteration 100. Averaged over 10 different runs per 1 active user. The x-axis represents the joke id.

with too many recommendations), FlockRecom computed better recommendations, suggesting a more effective and realistic recommendation strategy. Morever, FlockRecom is more successful in exploration and in overcoming over-specialization.

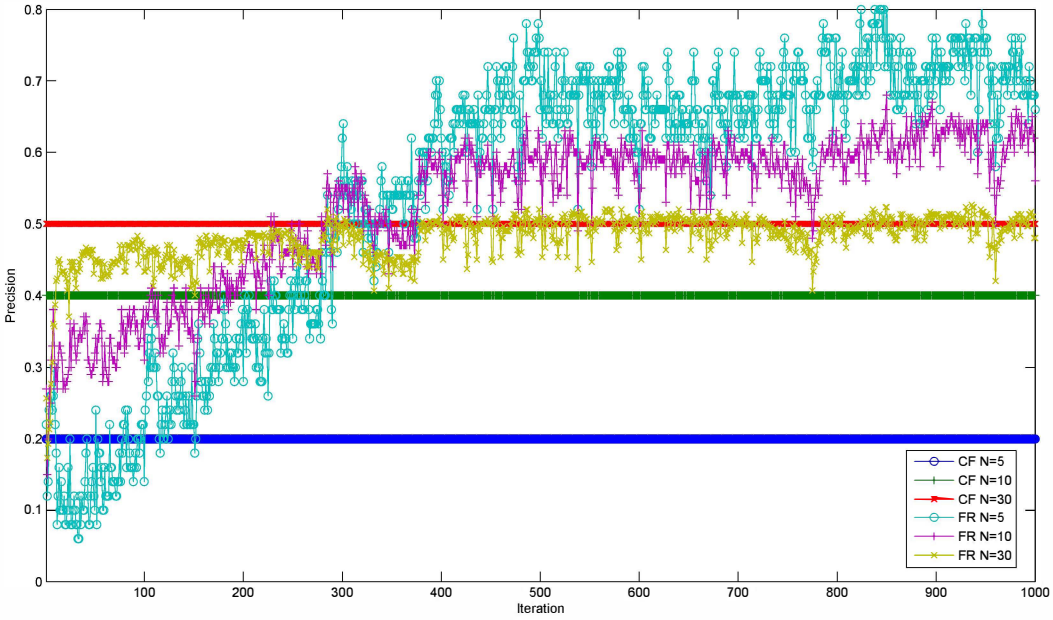

Fig. 6. Flock-based recommender system (FR) compared to standard collaborative filtering-based recommender system (CF) based on the average precision values for different numbers of top-n recommendations, over time. Averaged over 1 active user, 10 different runs. The x-axis represents the iteration number.

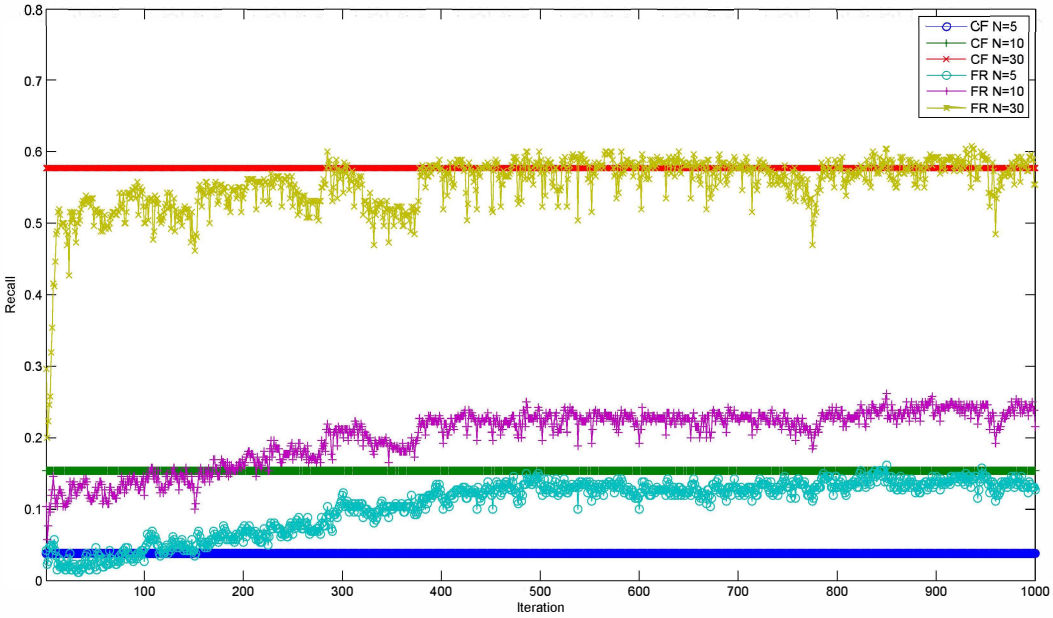

Fig. 7. Flock-based recommender system (FR) compared to standard collaborative filtering-based recommender system (CF) based on the average recall values for different numbers of top-n recommendations, over time. Averaged over 1 active user, 10 different runs. The x-axis represents the iteration number.

Figures 6 to 8 show the results for a typical active user. The figures display the quality versus time in comparison with the traditional collaborative filtering (CF). In Figure 6, plots labeled FR (FlockRecom) show that precision gets significantly improved in the first 400 iterations and later keeps increasing slowly. As expected, when a smaller number of items are recommended, precision was higher. Comparing FR to CF in Figure 6, we observe that precision values are better

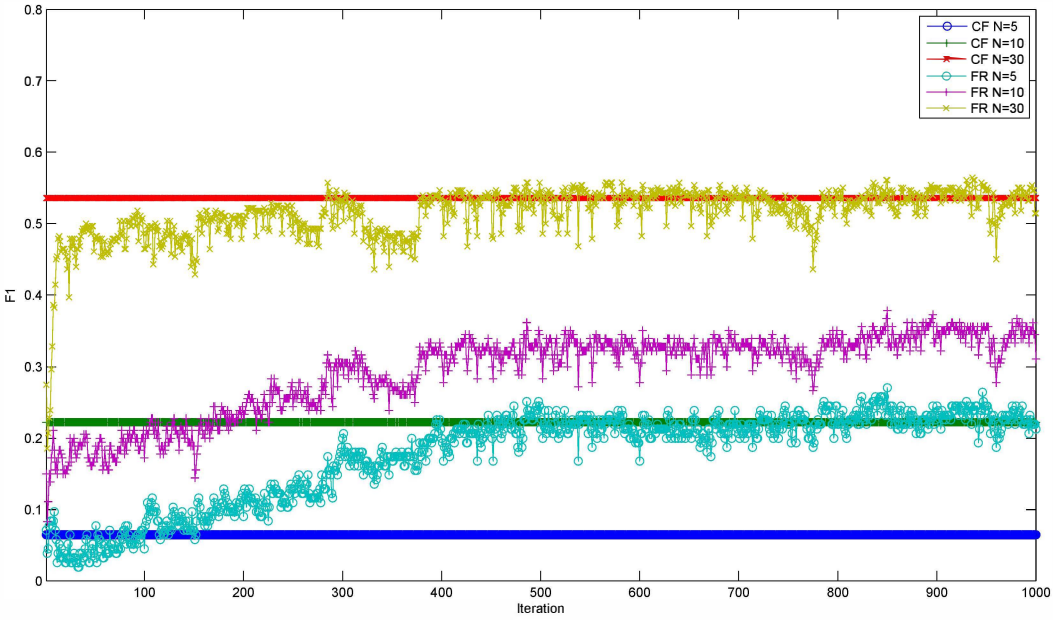

Fig. 8. Flock-based recommender system (FR) compared to standard collaborative filtering-based recommender system (CF) based on the average F1 values for different numbers of top-n recommendations, over time. Averaged over 1 active user, 10 different runs. The x-axis represents the iteration number.

for FR for small N, and similar for FlockRecom and CF for bigger N. Moreover, FlockRecom provides more variety in the recommendations. Similarly, in FlockRecom, recall was improved with the number of iterations, as seen in Figure 7. For small N, FlockRecom produced better results. For big N, FlockRecom needed more iterations to compute similar recall values to CF. For both FlockRecom and CF, recall values increased as N, the number of recommended items, was increased, as shown in Figure 7.

To sum up, Figure 6, Figure 7, and Figure 8 show that, as the number of iterations increases, the neighborhood quality increases, thus the quality of the recommendations increases. We notice that FlockRecom continues to improve its recommendations, eventually reaching higher quality levels compared to standard CF. After enough number of iterations, FlockRecom produced better results than CF. Moreover, FlockRecom was more successful at exploration and overcoming over-specialization. These improvements are hence due solely to the dynamic nature of the flocking behavior of the agents that form dynamic neighborhoods that do not cause stagnation in the recommendations.

Table II shows the quality levels over several iterations for the two methods for 3 values of N. Note how FlockRecom clearly outperforms CF by "continuing" to learn and thus improving recommendations with time.

VI. Conclusion

In this research, a new recommender system approach called the flocks-of-agents based recommender system (FlockRecom) was presented. This new approach is based on swarm intelligence, specifically, the dynamic collaboration between bird flocks in nature. The results were compared to the traditional user-based nearest neighbor collaborative filtering and FlockRecom was more successful at providing variety in the recommendations without losing recommendation quality. One problem suffered by some recommender systems is over-specialization. When the recommendations are limited to the user's behavior or user's profile, the user can be restricted to seeing only similar items, and there will be no randomness. In artificial intelligence, this problem is known as the exploration/exploitation dilemma. Although collaborative filtering can counteract over-specialization by suggesting different items, the dynamic structure of the FlockRecom algorithm makes it more successful at solving the exploration/exploitation dilemma, which is also practically observed in the experimental results. This and the dynamic nature of the algorithm suggests that the proposed nature inspired recommendation system looks very promising for dynamic environments, especially where a concept drift exists. Thus, more experiments on different datasets, especially ones that represent dynamic environments, are planned in the future.

References

[1] [Online]. Available: http://eigentaste.berkeley.edu/dataset/

[2] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web, LNCS 4321*, 2007, pp. 291–324.

[3] J. Schafer, J. Konstan, and J. Reidel, "Recommender systems in e-commerce," in *Proceedings of ACM Conference on E-commerce*, 1999, pp. 158–166.

TABLE II

THE QUALITY LEVELS AVERAGED OVER 10 RUNS OF 1 ACTIVE USER AT SEVERAL ITERATIONS FOR THE FLOCKRECOM AND CF AT 3 VALUES OF N.

FlockRecom						CF						
N=5		N=10		N=30		N=5		N=10		N=30		Iter.
Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	
0.28	0.05	0.42	0.16	0.46	0.54	0.20	0.04	0.4	0.15	0.5	0.58	200
0.62	0.12	0.57	0.22	0.50	0.58	0.20	0.04	0.4	0.15	0.5	0.58	400
0.84	0.16	0.68	0.26	0.52	0.60	0.20	0.04	0.4	0.15	0.5	0.58	850

[4] E. Saka and O. Nasraoui, "Improvements in flock-based collaborative clustering algorithms," in *Computational Intelligence Collaboration, Fusion and Emergence, Intelligent Systems Reference Library*, C. Mumford and L. Jain, Eds. Springer, 2009, pp. 639–672.

[5] E. Saka and O. Nasraoui, "Simultaneous clustering and visualization of web usage data using swarm-based intelligence," in *Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'08)*, 2008.

[6] F. Picarougne, H. Azzag, G. Venturini, and C. Guinot, "A new approach of data clustering using a flock of agents," *Evolutionary Computation*, vol. 15, no. 3, pp. 345–367, 2007.

[7] M. Pazzani and D. Billsus, "Learning and revising user profiles: The identification ofinteresting web sites," *Mach. Learn.*, vol. 27, no. 3, pp. 313–331, 1997.

[8] M. Balabanović, "An adaptive web page recommendation service," in *AGENTS '97: Proceedings of the first international conference on Autonomous agents*. New York, NY, USA: ACM, 1997, pp. 378–385.

[9] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens: Applying collaborative filtering to usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[10] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl, "Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system," in *In Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, Seattle, Washington, 1998, pp. 345–354.

[11] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

[12] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," *Artificial Intelligence Review*, vol. 13(5-6), pp. 393–408, 1999.

[13] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[14] F. Heppner and U. Grenander, "A stochastic nonlinear model for coordinated bird flocks," in *The Ubiquity of Chaos*, S. Krasner, Ed. Washington: AAAS, 1990, pp. 233–238.

[15] I. D. Couzin, J. E. N. S. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective memory and spatial sorting in animal groups," *Journal of Theoretical Biology*, vol. 218, no. 1, pp. 1–11, September 2002.

[16] G. Proctor and C. Winter, "Information flocking: Data visualisation in virtual worlds using emergent behaviours," *Lecture Notes in Computer Science*, vol. 1434, pp. 168–176, 1998.

[17] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[18] C. J. Rijsbergen, *Information Retrieval*, 22nd ed. Butterworth-Heinemann, 1979.

[19] R. Belew, "Adaptive information retrieval: using a connectionist representation to retrieve and learn about documents," in *Proceedings of the 12th annual international ACM SIGIR conference on Research and development in information retrieval*, Cambridge, Massachusetts, United States, 1989, pp. 11 – 20.

[20] R. K. Belew, *Finding out about: a cognitive perspective on search engine technology and the WWW*. New York, NY, USA: Cambridge University Press, 2000.