A Scalable People-to-People Hybrid Reciprocal Recommender Using Hidden Markov Models

Ammar Alanazi[*] Michael Bain[†]

Abstract
Most existing reciprocal recommender systems use either profile similarity or interaction similarity to recommend new matches, assuming that user preferences are static and ignoring temporal aspects of user behaviour. This paper takes a different approach, and addresses the issue of representing user preferences as dynamic. We introduce a new representation for changes in user preferences and use that representation in creating a reciprocal recommender system applied to online dating.

In this paper, we develop a general framework for combining a Hidden Markov model (HMM) content-based reciprocal recommender system with collaborative filtering techniques to create a unified hybrid recommender. Additionally, a new similarity measure is introduced to rank the recommendations generated by this hybrid recommender. Moreover, we propose, design and implement a reciprocal recommender system using the suggested framework and the new similarity measure. Evaluation of this system shows that it generates better recommendations than existing systems in a time-efficient manner.

1 Introduction

Most of the existing work on recommender systems is built on the assumption that users' behaviours are static and do not change over time. More recently, this assumption has begun to be relaxed in work on temporal recommendation [20, 22, 13]. However, to the best of our knowledge no work has addressed the problem of temporal *reciprocal* recommendation, such as occurs in the context of online dating, employment websites, and other people-to-people interactions.

Analysis of real-world data from a dating website on people-to-people interactions [2] shows that people's behaviour and activity levels do change over time, which leads to the conclusion that we need a dynamic model to generate better recommendations.

To capture these temporal changes, in this paper we describe a scalable hybrid Hidden Markov Model [19] reciprocal recommender system that captures how each user's behaviour evolves over time and generates recommendations accordingly. The proposed hybrid recommender combines collaborative filtering techniques with the HMM recommender proposed in [3] to generate recommendations.

In this paper we introduce a novel hybrid recommender system that combines the ability of collaborative filtering methods to generate recommendations with the high performance of the predictions of the Hidden Markov model recommender that was proposed in [3], to generate recommendations in reciprocal domains with high success rates. The Collaborative Filtering Hidden Markov Models Hybrid Recommender ***CFHMM-HR*** was tested on an industrial-scale data that was acquired from a real dating website and the results show that CFHMM-HR outperforms its counterparts.

Moreover, CFHMM-HR is a scalable hybrid recommender that can run in real-world applications. Although CFHMM-HR was only tested on an online-dating dataset, we believe that the same model can be applied in other reciprocal domains after a few changes in the preprocessing level.

The key contribution we present in this paper is a novel hybrid recommender that combines techniques from content-based recommender systems and collaborative filtering recommender systems to generate recommendations in a people-to-people domain and which outperforms the previous recommender systems that were reported on a similar domain. In particular, the proposed hybrid recommender offers the following novel features:

- It presents a novel hybrid recommender that generates recommendations with high success rate in comparison to its counterparts.
- It presents a recommender that can work on industrial sized datasets in a timely manner and that can be deployed online.
- It introduces a new similarity measure that minimises the number of false positives and maximises the number of true positives predicted by the recommender. Consequently, the new similarity measure maximises the success rate of the recommendations generated by the model.

[*]King Abdulaziz City for Science and Technology, National Center for Computer Technology and Applied Math, Riyadh, Saudi Arabia
[†]The University of New South Wales, Faculty of Engineering, Sydney, Australia

2 Related Work

Although recommender systems have been investigated thoroughly in the literature, research on temporal aspects of the recommendation problem has only attracted attention in the past few years. Most of the time-aware recommenders proposed in the literature [13, 20, 22, 11, 26, 10, 12] are variations of the Matrix Factorisation (MF) model. However, initial experimentations on MF in people-to-people reciprocal recommendation did not work well [23].

The area of people-to-people reciprocal recommenders has attracted little research attention in comparison to traditional item-to-user recommenders. Although some interesting models were proposed in the literature such as [1, 14, 8, 23, 15, 7, 24, 17], to the best of our knowledge no one has addressed the temporal aspect of the problem explicitly.

3 Dataset

The dataset used to test the model is a real-world commercial dataset from a dating website. In the dating domain, there are users who initiate interactions, we call them *senders*, and people who receive interactions, and we call them *recipients*. Senders and recipients can overlap which means a user can be a recipient and a sender at the same time. There are different forms of interactions that can be exchanged such as predefined messages, emails and chats. In this research, we use the predefined messages, we call them *messages*, to train and test our model because this is the first method of communication between users in most cases and depending on the success of these messages, users can further their communications and exchange other forms of interactions.

When a predefined message is sent, the recipient can ignore this message and not reply to it, reply with a positive predefined message or reply with a negative one. We have only considered messages that have replies to them and classify them as positive or negative interactions based on the reply message.

The dataset has over 3 million users and over 80 million interactions exchanged between these users. Therefore, using the whole dataset is not feasible and representative subsets have to be used instead. To generate training and testing data for our model, a time period was randomly selected (e.g. from March 1^{st} to March 15^{th}, 2009) and all active users during this time period were used as the experiment population. Then, for users in the selected population, all their interactions, even interactions outside the selected time period, were obtained and used to build the model. Several populations were generated and average results across these populations will be presented later in this paper.

The average population size is over 195,000 users of which a little over 16,000 were recipients and about 190,000 were senders. These users exchanged over two million messages amongst themselves with an average baseline success rate of 15%. Each population was divided into 70% training data and 30% test data.

This dataset was chosen because it is a real-world commercial reciprocal dataset that has temporal dynamics. Although users' life cycles are mostly short in a dating website [2], there are several life-cycle phases to capture and these changes between phases have their effects on the decision of initiating an interaction and the decision of accepting one.

4 Evaluation Metrics

In applications that require generating actual recommendations to users, we are interested in measuring how many of these predictions will be used [21, 4, 9]. Dating is one of these applications.

In this paper we use the following two metrics that measure the *usage* of the recommendations: ***Success Rate***, which is defined as the proportion of generated recommendations that are correct and ***Recall***, which is the proportion of successful interactions in the test set that was predicted successfully by the model [18].

More formally, let R be the set of recommendations generated by the model, R_+ be the subset of R that is correct, I_+ be the set of successful interactions in the test data and $Size(S)$ is the size of a set S.

$$SuccessRate = \frac{Size(R_+)}{Size(R)} = \frac{TP}{TP + FP}$$

$$Recall = \frac{Size(R_+)}{Size(I_+)} = \frac{TP}{TP + FN}$$

Generally, recommender systems in online dating applications are required to generate a pre-determined number of recommendations and in this case the most important evaluation metric is Success Rate [21]. However, since experiments in this paper are all offline experiments that are performed on historical data, we will present Recall values as well, but the focus of this research is to improve the Success Rate.

Additionally, we will use the ***F-measure***, which is a weighted average of success rate and recall [18]. The general formula to calculate F-measure is:

$$F_\beta = (1+\beta^2).\frac{SuccessRate.Recall}{(\beta^2.SuccessRate) + Recall} \text{where} \quad \beta > 0$$

When $\beta = 1$, similar weights are given to both success rate and recall. We decided to use $\beta = 0.25$ to put more emphasis on success rate since it is the focus of this research.

5 A Hidden Markov Models Content-Based Recommender

Most of the existing recommendation algorithms deal with the recommendation problem as a two-state problem, in which all historical data is considered as one state and the recommendation problem becomes predicting the next state. In this research, we propose a model that considers the temporal aspects of the problem and utilises them to better personalise the recommendations. We track the changes of the graph over multiple time periods and observe how it evolves over time then use this gained knowledge to predict the future graph.

5.1 Design Hidden Markov models have been proven to work successfully in a wide range of applications that require a temporal model with the capabilities of recognising sequences [20, 5]. Therefore, we decided that HMMs will be our model of choice to represent the dynamically changed user preference.

To track the changes of user preference over multiple time periods, we present here a novel representation for interactions. That is, we represent each interaction I_k in the data as a sequence of size n as follows:

$$\begin{aligned} & I_k = (O_{k-n}, O_{k-n+1}, \ldots, O_{k-2}, O_{k-1}, O_k) \\ & \quad \text{if} \quad k \geq n \\ \text{or} \quad & I_k = (\phi, \phi, \ldots, O_0, \ldots, O_{k-2}, O_{k-1}, O_k) \\ & \quad \text{if} \quad k < n \end{aligned}$$

where O_k is the k_{th} observation vector.

Each observation vector represents a message in the dataset and it consists of:

- A selected set of ***profile data*** for the sender and the recipient: gender, age, location, marital status, sexuality, number of profile photos, body type, height, occupation industry, occupation level and diet.
- A set of ***derived data***: the difference in age between the sender and the recipient and the physical distance between them (in kilometres).
- A set of ***temporal data***: activity in the last 7 days, activity in the last 28 days and number of days since receiving the previous message. Activity here is defined as the difference between received messages and sent replies and these features are calculated for the recipient only.

Another way to describe the suggested new representation of interactions is, instead of considering the current message ($Message_k$) as an isolated observation (i.e. $Message_k$ is represented as O_k), we consider it as a sequence of events leading to O_k taking place (i.e. $Message_k$ is represented as I_k). Each interaction is then classified to a successful or a failed interaction. The interaction is deemed successful if $Message_k$ received a positive reply. Otherwise, the interaction is classified as a failed one in the cases of no-reply or a negative reply.

Following that, we designed a HMM system that we can train using a subset of the dataset (training data) and then use the resulted model as a recommender. More details and experimental results of this model can be found in [3].

6 A Hybrid Recommender Combining Collaborative Filtering and Hidden Markov Models

6.1 Limitations of the Content-Based HMM Recommender Although the experimental results of the content-based HMM recommender [3] are promising, there is one main limitation to that model. The only way of actually generating recommendations is to use brute-force to generate all the possible interactions that could occur and then pass them through the model to predict whether they will succeed or fail. However, this is not feasible due to the size of the dataset. Even with a small sample of 1,000 recipients, there is an average of 12,000 senders interacting with these recipients. Consequently, brute-force will produce 12,000,000 interactions to be validated and this will not work in a timely manner in a real-world recommender. Moreover, that sample is too small to be representative of the whole dataset.

6.2 Design One of content-based recommenders' strengths is that they can be used as filters on recommendations generated by other methods [16]. On the other hand, collaborative filtering recommenders have the ability to generate a list of recommendations by utilising the similarities between the users.

Therefore, to overcome the limitations of the HMM recommender and be able to generate recommendations, we decided to use a collaborative filtering recommender first to generate the recommendations. Then, we test the top N recommendations using the HMM model and filter out the unsuccessful predictions.

The CFHMM-HR model works in the following order (Figure 1):

- Training data is used to train the HMM recommender and used by the collaborative filtering recommender to generate the initial list of recommendations.
- The initial list of recommendations gets validated

by the HMM recommender. The output of this step is another list, the second list, of recommendations which is a smaller subset of the initial list.

- The second list of recommendations gets ranked using a combination measure of likelihood and collaborative filtering similarity and the final list of recommendations is generated.

To represent the messages of the initial list of recommendations as interactions (see above), we assume that each one of these messages are received immediately after the last message of the training data for each user. Formally:

$$\begin{aligned} & I_r = (O_{tr_{k-n+1}}, O_{tr_{k-n+2}}, \ldots, O_{tr_k}, O_r) \\ & \quad \text{if} \quad k \geq n-1 \\ \text{or} \quad & I_r = (\phi, \phi, \ldots, O_0, \ldots, O_{tr_{k-1}}, O_{tr_k}, O_r) \\ & \quad \text{if} \quad k < n-1 \end{aligned}$$

where tr_k is the last message in the training data for each user, r is the recommended message, O_x is the observation vector for the message x and I_x is the message x represented as an interaction.

The model is built so that it can work with any collaborative filtering recommender as long as it generates the initial list of recommendations in a compatible syntax. For the collaborative filtering part of the recommender, we experimented with three different models: Basic CF+ [14], SIM-CF [23] and ProCF [8]. These models were selected because SIM-CF [23] is the model that was chosen to be the recommender for the dating website that we obtained its dataset for this research and SIM-CF's main strength is its high recall. ProCF [8] is one of the best performing models, success rate wise, reported on the same dataset. Finally, Basic CF+ [14] is a simple model that implements the basic idea of collaborative filtering in an easy to understand way.

In each experiment, the HMM part of CFHMM-HR receives a list of the top 200 candidates for each user from the CF recommender. The HMM recommender then filters and re-ranks that list to get the top 50.

Since CFHMM-HR receives the initial list of recommendations with their similarity scores from a CF recommender and we have no control over the similarity measure used in that CF recommender, it is not possible to derive a combined similarity measure from first principles. Instead, we derive a new heuristic similarity measure that combines the similarity score received from the CF recommender with the likelihood values generated by the HMM recommender. The new similarity measure was engineered to assure that each element of the measure has the required effect.

The final list of recommendations generated by CFHMM-HR is ranked based on this new heuristic similarity measure we call $HMMSIM$. The formula to calculate $HMMSIM$ is as follows:

$$HMMSIM(m_i) = \Delta_{likelihood}(m_i) + sim(m_i) + \alpha$$

where:

$$\alpha = \begin{cases} \text{large positive constant,} & \text{if} \quad \Delta_{likelihood}(m_i) \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

and:

$$\Delta_{likelihood}(m_i) = likelihood_{success}(m_i) - likelihood_{failure}(m_i)$$

Also, $sim(m_i)$ is the similarity, between m_i's sender and m_i's recipient, generated by the CF recommender for the message m_i

One of the findings we discovered during the experimentations on the HMM content-based recommender [3] is that it is a conservative recommender that tends to under-predict in most cases. However, its predictions are highly reliable. On the other hand, we noticed that the CF recommenders tend to over-predict both true positives and false positives. Therefore, the main effect of adding $\Delta_{likelihood}(m_i)$ in $HMMSIM$ is to minimise the number of false positives predicted by $sim(m_i)$ by demoting their score.

Further, since the predictions of the HMM recommender are highly reliable, we want the messages that are predicted to be successful by it to be on the top of the list. Therefore, we add the large constant α to promote these messages to the top of the list ($\Delta_{likelihood}(m_i) \geq 0$ means m_i is predicted to be successful).

Additionally, we find that $\Delta_{likelihood}(m_i)$ ranges between -15 and +15, while $sim(m_i)$ value can be over 180 for popular users. Which means that in such cases of popular users with high $sim(m_i)$ values, the addition of $\Delta_{likelihood}(m_i)$ will not be sufficient to have the required effect of demoting the overall score $HMMSIM$. Therefore, the addition of α balances the weights of $sim(m_i)$ and $\Delta_{likelihood}(m_i)$ as well.

6.3 Implementation The Hidden Markov models Toolkit (HTK) [25] was used to implement the content-based Hidden Markov models part of the recommender and Java was used to implement all the preprocessing and post-processing tools.

7 Experimental Results

This proposed model was also evaluated using the same dataset described in Section 3. Users of the website

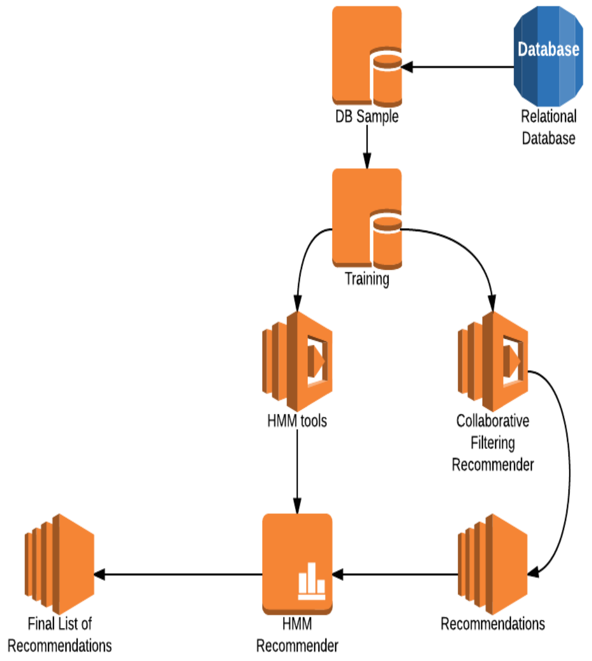

Figure 1: The framework for CFHMM-HR.

did not have access to recommendations generated by our model and therefore the model was evaluated using historical data.

7.1 CFHMM-HR with Basic CF+ In this experiment we used the Basic CF+ [14] as the CF recommender. Comparing the accuracy of the recommendations between Basic CF+ alone and CFHMM-HR with Basic CF+ as the CF recommender shows a noticeable overall improvement. The success rate of CFHMM-HR for top 50 recommendations is 15.5% more than the success rate of Basic CF+ and for the top 10 recommendations it is 18.2% more. The recall of the top 50 recommendations is 0.9% more in CFHMM-HR while the recall of the top 10 recommendations is the same in both cases. Please refer to Table 1 for detailed results.

7.2 CFHMM-HR with SIM-CF This section presents the experimental evaluation of CFHMM-HR using SIM-CF [23] as the CF recommender. There is an improvement in the success rate and a slight loss of recall. In the top 50 recommendations, CFHMM-HR improves the success rate by 5.7% while losing 0.6% recall. Similarly, in the top 10 recommendations there is an improvement of 5.1% in success rate and a setback of 2.9% in recall when using the CFHMM-HR instead of using the SIM-CF alone. Using the $F_{0.25}$ score to evaluate the overall performance, CFHMM-HR outperforms SIM-CF in all top ranks. More detailed results are presented in Table 2.

Table 1: A comparison between the results of Basic CF+ and the result of CFHMM-HR ($n = 5$).

Basic CF+			
Rank	Success Rate	Recall	$F_{0.25}$ score
top 10	0.252	0.034	0.183
top 20	0.235	0.057	0.199
top 30	0.226	0.076	0.202
top 40	0.219	0.092	0.203
top 50	0.213	0.105	0.201

CFHMM-HR			
Rank	Success Rate	Recall	$F_{0.25}$ score
top 10	0.434	0.034	0.256
top 20	0.406	0.060	0.303
top 30	0.388	0.082	0.318
top 40	0.376	0.099	0.323
top 50	0.368	0.114	0.325

Table 2: A comparison between the results of SIM-CF and the result of CFHMM-HR ($n = 5$).

SIM-CF			
Rank	Success Rate	Recall	$F_{0.25}$ score
top 10	0.296	0.107	0.268
top 20	0.279	0.163	0.268
top 30	0.265	0.202	0.261
top 40	0.256	0.234	0.255
top 50	0.250	0.260	0.250

CFHMM-HR			
Rank	Success Rate	Recall	$F_{0.25}$ score
top 10	0.347	0.078	0.289
top 20	0.341	0.146	0.316
top 30	0.327	0.191	0.314
top 40	0.314	0.223	0.307
top 50	0.307	0.254	0.303

7.3 CFHMM-HR with ProCF Here we present the results of CFHMM-HR with ProCF [8] as the CF part of it. We notice an improvement in success rate and a small drop in recall for the top 30 ranks and below. For the top 50 recommendations, CFHMM-HR has 18.9% more success rate than ProCF alone and 0.5% more recall. For the top 10 recommendations, CFHMM-HR shows a 20.2% increase in success rate and a 0.3% decrease in recall. Using the $F_{0.25}$ score to evaluate the overall performance, CFHMM-HR outperforms ProCF in all top ranks. Please refer to Table 3 for more results.

Table 3: A comparison between the results of ProCF and the result of CFHMM-HR ($n = 5$).

ProCF				CFHMM-HR			
Rank	Success Rate	Recall	$F_{0.25}$ score	Rank	Success Rate	Recall	$F_{0.25}$ score
top 10	0.485	0.032	0.266	top 10	0.687	0.029	0.294
top 20	0.471	0.048	0.310	top 20	0.667	0.042	0.354
top 30	0.455	0.059	0.326	top 30	0.653	0.058	0.407
top 40	0.448	0.068	0.337	top 40	0.642	0.070	0.434
top 50	0.440	0.074	0.341	top 50	0.629	0.079	0.447

8 Analysis of Run-Time

Although the main focus of this research is to find a recommendation model that identifies temporal changes and utilises them to better understands user behaviour, there is another aim that is of similar importance. The other aim is that the final model is scalable, time-efficient and can be deployed in a real-world application.

To demonstrate the efficiency of CFHMM-HR, we have calculated the run-time for every step the recommender performs over a large number of experiments. These experiments were run on a Windows Server 2008 machine with the following specifications: Intel Xeon CPU, 2.80 GHz, 4 processors, 32 GB RAM.

The typical scenario for running CFHMM-HR online is as follows:

1. Train the HMM part of CFHMM-HR on a large training data sample. This step can be performed offline and there is no need for frequent re-training. We believe that repeating this step once every month would suffice.

2. Train the CF part of CFHMM-HR overnight. This step needs to be repeated more frequently.

3. Generate the initial list of recommendations from the CF recommender.

4. Test, filter and re-rank the initial list of recommendations using the latest trained HMM recommender. This step needs to be performed overnight and repeated more frequently as well.

5. Send the final list of recommendations to users.

The run-time for training the HMM recommender is shown in Table 4. The average time it takes per message is 106 milliseconds and we suggest using a sufficiently large representative training sample (1-2 million messages). 94% of the time taken in training is spent by the HTK tools that were used to implement the HMM recommender and the remaining 6% is spent by the Java software tool. Moreover, of the time spent by the Java software tool, 86% was spent in input/output operations to process the HTK formatted files. We cannot estimate the proportion of time spent in input/output operation by the HTK tools but we believe it is similar to the Java software tool. The HTK toolkit requires the data to be stored in specially formatted files on the hard disk drive (HDD) which adds a time overhead for opening these files and reading the data and in this case the overhead consumes about 80% of the total time. To deploy CFHMM-HR on an online application, we recommend developing alternative HMM software tools that read the data directly from the DBMS or from memory to eliminate the time overhead.

The run-time of the CF part of CFHMM-HR varies depending on the algorithm used. The three algorithms that were used in this research have been tested in an online trial and they run efficiently [23].

Testing the initial recommendation list by the HMM recommender run-time is presented in Table 5. It takes an average of 8.8 milliseconds to test one message of which 20% is spent by the HTK tools and the remaining 80% is spent by the Java software tool. About 88% of the Java software tool time is spent on input/output operations to generate the HTK formatted files. This leads us to the same conclusion that developing alternative HMM software tools that read the data directly from the DBMS or from memory would minimise the run-time significantly.

9 Discussion

CFHMM-HR is a general framework for combining CF techniques with the HMM content-based recommender in a two-step hybrid recommender. Such a combination overcomes one of the main shortcomings of the HMM recommender [3] which is its inability to generate actual recommendations. It also improves the success rate of the CF recommenders considerably. However, the improvement of CFHMM-HR depends on the initial list of recommendations generated by the CF recommender. For example, the best success rate for CFHMM-HR is when it is combined with ProCF and that is because of the three CF methods we combined with the HMM CB recommender, ProCF is the one with the highest success

Table 4: Average run-time for training the HMM recommender (per message).

	Total	HTK Tools	Java Software Tool		
			DB Operations	I/O Operations	Other
Time (msec)	106.131	99.259	0.969	5.894	0.009
Percentage		93.52%	0.91%	5.55%	0.01%

Table 5: Average run-time for HMM recommender testing (per message).

	Total	HTK Tools	Java Software Tool		
			DB Operations	I/O Operations	Other
Time (msec)	8.787	1.800	0.761	6.213	0.013
Percentage		20.49%	8.66%	70.71%	0.14%

rate. On the other hand, the best recall for CFHMM-HR is when it is combined with SIM-CF because it is the model with the highest recall. Nevertheless, in all experimentations performed in this research, CFHMM-HR outperforms the use of the CF recommender alone in success rate.

However, while it is possible to deploy CFHMM-HR in an online application in its current setup, we believe that this is not the ideal setup and more optimisation is needed. Mainly, we suggest developing alternative HMM software tools instead of using the HTK toolkit. Because HTK is designed to read data from files stored on HDD, a large proportion of the run-time of CFHMM-HR is spent on writing and reading these files. We estimate that the time overhead caused by the I/O operations is about 80% of the reported run-time and we recommend developing HMM tools that read the data directly from the DBMS or from memory.

In the domain of online dating the interaction vectors of users tend to be sparse and short. Initial experimentations of collaborative filtering methods that are popular with researchers, such as matrix factorization MF, did not work well [23]. A possible reason for that is that in the domain of online dating the interaction matrix is Boolean (positive or negative) and in such cases applying matrix factorisation is harder than applying it on a numerical matrix (ratings) [6]. Therefore, we compared our model to other models that are known to work on such data.

10 Conclusion and Future Work

In this paper we presented a hybrid model for people-to-people recommendations using HMM that can capture the temporal changes of users' behaviours and generates better personalised recommendations based on this. Evaluating this model using a commercial dataset for a dating website shows a significant improvement in the success rate of recommendations.

The model combines a HMM content-based recommender and a collaborative filtering algorithm to generate recommendations. In future, we plan on using other dynamic models to represent the recommender such as coupled Hidden Markov models or the Collaborative Kalman Filtering model [22]. Moreover, in CFHMM-HR, the recommendations are generated by a reciprocal CF method and then re-ranked using the temporal HMM content-based recommender. Incorporating the temporal dynamics in the CF part of the recommender as well would be an interesting extension to our model.

Acknowledgments

We would like to thank Smart Services Cooperative Research Centre and their industrial partners for providing the datasets.

References

[1] J. Akehurst, I. Koprinska, K. Yacef, L. Pizzato, J. Kay, and T. Rej. CCR: A content-collaborative reciprocal recommender for online dating. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence - Volume 3*, IJCAI'11, pages 2199–2204. AAAI Press, 2011.

[2] A. Alanazi and M. Bain. Ranking interaction-based collaborative filtering recommendations using temporal features in online dating. In K. Soliman, editor, *Innovation and Sustainable Competitive Advantage: From Regional Development to World Economies*, volume 1-5, pages 450–457. Int Business Information Management Assoc-IBIMA, 2012. 18th IBIMA Conference, Istanbul, TURKEY, MAY 09-10, 2012.

[3] A. Alanazi and M. Bain. A people-to-people content-based reciprocal recommender using Hidden Markov models. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 303–306, New York, NY, USA, 2013. ACM.

[4] A. Bellogin, P. Castells, and I. Cantador. Precision-oriented evaluation of recommender systems: An algo-

rithmic comparison. In *Proceedings of the 5th ACM Conference on Recommender Systems*, RecSys '11, pages 333–336, New York, NY, USA, 2011. ACM.

[5] J. A. Bilmes. What hmms can do. *IEICE Transactions on Information and Systems*, E89-D(3):869–891, March 2006.

[6] W. Buntine and A. Jakulin. Discrete component analysis. In C. Saunders, M. Grobelnik, S. Gunn, and J. Shawe-Taylor, editors, *Subspace, Latent Structure and Feature Selection*, volume 3940 of *Lecture Notes in Computer Science*, pages 1–33. Springer Berlin Heidelberg, 2006.

[7] X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y. Kim, P. Compton, and A. Mahidadia. Collaborative filtering for people to people recommendation in social networks. In J. Li, editor, *AI 2010: Advances in Artificial Intelligence*, volume 6464 of *Lecture Notes in Computer Science*, pages 476–485. Springer Berlin Heidelberg, 2011.

[8] X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y. Kim, P. Compton, and A. Mahidadia. ProCF: Probabilistic collaborative filtering for reciprocal recommendation. In J. Pei, V. Tseng, L. Cao, H. Motoda, and G. Xu, editors, *Advances in Knowledge Discovery and Data Mining*, volume 7819 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2013.

[9] P. G. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1-2):67–119, Feb. 2014.

[10] F. C. T. Chua, R. J. Oentaryo, and E. Lim. Modeling temporal adoptions using dynamic matrix factorization. In *IEEE 13th International Conference on Data Mining*, number Dallas,, pages 91–100, TX, USA, December 2013.

[11] S. Gultekin and J. Paisley. A collaborative kalman filter for time-evolving dyadic processes. In *Proceedings of the IEEE International Conference on Data Mining*, ICDM '14, pages 140–149, Washington, DC, USA, 2014. IEEE Computer Society.

[12] B. Ju, Y. Qian, M. Ye, R. Ni, and C. Zhu. Using dynamic multi-task non-negative matrix factorization to detect the evolution of user preferences in collaborative filtering. *PLoS ONE*, 10(8):e0135090, 08 2015.

[13] Y. Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, April 2010.

[14] A. Krzywicki, W. Wobcke, X. Cai, A. Mahidadia, M. Bain, P. Compton, and Y. Kim. Interaction-based collaborative filtering methods for recommendation in online dating. In L. Chen, P. Triantafillou, and T. Suel, editors, *Web Information Systems Engineering, AI WISE 2010*, volume 6488 of *Lecture Notes in Computer Science*, pages 342–356. Springer Berlin / Heidelberg, 2010.

[15] L. Li and T. Li. MEET: A generalized framework for reciprocal recommender systems. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 35–44, New York, NY, USA, 2012. ACM.

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

[17] L. Pizzato, T. Rej, T. Chung, I. Koprinska, and J. Kay. RECON: a reciprocal recommender for online dating. In *Proceedings of the 4th ACM conference on Recommender systems*, pages 207–214. ACM, 2010.

[18] D. M. W. Powers. Evaluation: from precision, recall and f-measure to ROC, informedness, markedness and correlation. *International Journal of Machine Learning Technology*, 2(1):37–63, 2011.

[19] L. Rabiner. A tutorial on Hidden Markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.

[20] N. Sahoo, P. V. Singh, and T. Mukhopadhyay. A Hidden Markov model for collaborative filtering. *MIS Q.*, 36(4):1329–1356, Dec. 2012.

[21] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011.

[22] J. Sun, D. Parthasarathy, and K. Varshney. Collaborative kalman filtering for dynamic matrix factorization. *IEEE Transactions on Signal Processing*, 62(14):3499–3509, July 2014.

[23] W. Wobcke, A. Krzywicki, Y. S. Kim, X. Cai, M. Bain, P. Compton, and A. Mahidadia. A deployed people-to-people recommender system in online dating. *AI MAGAZINE*, 36(3):5–18, January 2015.

[24] P. Xia, B. Liu, Y. Sun, and C. Chen. Reciprocal recommendation system for online dating. In *Proceedings of 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Paris, France, 2015.

[25] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. A. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. *The HTK Book*. Cambridge University Engineering Department, March 2009.

[26] C. Zhang, K. Wang, H. Yu, J. Sun, and E.-P. Lim. Latent factor transition for dynamic collaborative filtering. In *Proceedings of the SIAM International Conference on Data Mining*, pages 452–460. Citeseer, 2014.