The Recommendation Game [*]

Using a Game-with-a-Purpose to Generate Recommendation Data

Sam Banks[†] , Rachael Rafter, Barry Smyth
Insight Centre for Data Analytics
School of Computer Science
University College Dublin
Dublin, Ireland
firstname.lastname@insight-centre.org

ABSTRACT

This paper describes a casual Facebook game to capture recommendation data as a side-effect of gameplay. We show how this data can be used to make successful recommendations as part of a live-user trial.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Filtering; H.3.5 [**Online Information Services**]: Web-Based Services

Keywords

Recommender Systems; Games-with-a-Purpose; Human Computation; Crowdsourcing

1. INTRODUCTION

Recommender systems suggest items to users. Most rely on user preferences, such as past ratings [1]. Many use matrix factorization methods to find hidden patterns within these preferences [9]. Yet others harness inter-user similarity to identify groups of like-minded users [2]. And some leverage social network data to infer trust relationships between users [6]. Collecting this information at scale is the subject of much research. Many approaches have been considered, from using explicit feedback such as transaction histories to inferring interest from implicit signals such as read-times or sharing [8]. More recently, crowdsourcing ideas have been considered (e.g. [11]); see also *CrowdRec* workshops[1]. In this paper we consider a crowdsourcing approach by developing a so-called game-with-a-purpose (GWAP) for soliciting recommendation data as a by-product of gameplay.

GWAPs are games that are simple and fun to play with gameplay contributing to some secondary problem solving goal; e.g. the ESP Game, arguably the original of the species, invites pairs of players to guess words for images [15] to improve indexing in image search. Other games have been developed to index audio and video [5, 10, 14]. GWAPs have also been developed for object segmentation [12] and even protein folding [3]. The power of a GWAP stems from its ability to attract many players who collectively contribute to a greater goal through their gameplay. Can GWAPs also be used for recommender systems? This question has been asked by [16] in the context of the Curator system, a game-with-a-purpose for recommending collections of items that go together; see also [4, 7]. In this paper we focus on a complementary set of recommendation matters related to user-user tie strength and user-item relevance.

2. THE RECOMMENDATION GAME

Our game is a Facebook app wherein a player p is tasked with matching a set of movies with their friends $Friends(p)$. We know movies each friend likes ($Likes(f)$) from their FaceBook 'likes'. Figure 1 shows the game in action. Movie posters float across the screen, becoming more erratic as the game progresses. By dragging a movie m to a friend $f \in Friends(p)$, player p *matches* m with f ($match(p, m, f)$) if they believe that f will like m. The more matches a player generates the more we learn their beliefs about the preferences of their friends; the set of matches that p generates is $Matches(p)$ (Equation 1). But how can we decide if these matches are correct? And, if we can decide, then doesn't this confirm recommendation data that we already know?

$$Matches(p) = \bigcup_{\forall f \in Friends(p)} \{(p, m, f) : match(p, m, f)\} \tag{1}$$

2.1 Resolving the Matching Paradox

For a match we either know that f likes m (a *known match*) or we have no such knowledge (an *unknown match*). Either way, we can infer recommendation data as follows.

2.1.1 From Known Matches to Tie Strength

In the case that f is known to like m ($m \in Likes(f)$) then we learn something about p's understanding of f's (movie)

[*] This work is supported by Science Foundation Ireland through the Insight Centre for Data Analytics under grant number SFI/12/RC/2289.

[†] Sam Banks completed this work as a final-year Computer Science student in University College Dublin.

[1] http://crowdrecworkshop.org/

RecSys'15 September 16–20, 2015, Vienna, Austria.
© 2015 ACM. ISBN 978-1-4503-3692-5/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2792838.2799675.

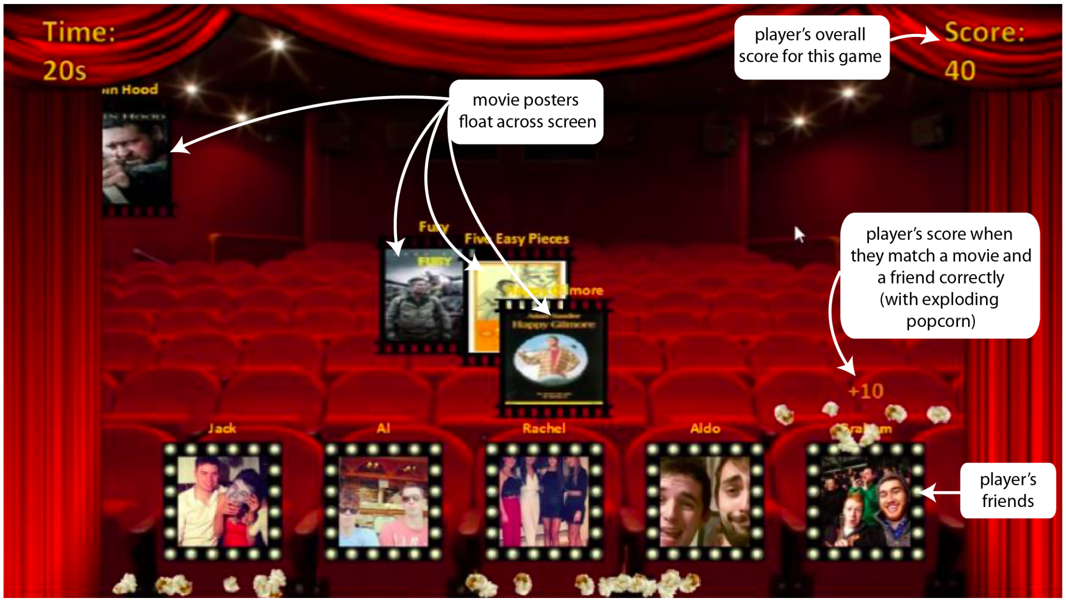
Time:
20s
Score:
40
player's overall score for this game
movie posters float across screen
Robin Hood
Fury
Five Easy Pieces
Happy Gilmore
player's score when they match a movie and a friend correctly (with exploding popcorn)
+10
Jack
Al
Rachel
Aldo
Graham
player's friends

Figure 1: The Recommendation Game in action.

interests. The more such known matches, the better p seems to know f; useful recommendation data in itself. For example, we can estimate this as the proportion of known matches that p generates for f relative to all matches p generates for f in a given set of games; see Equations 2 – 4.

$$\begin{aligned} &FriendMatches(p, f) = \\ &\qquad \{(p', m', f') \in Matches(p) : p = p' \wedge f = f'\} \end{aligned} \tag{2}$$

$$\begin{aligned} &KnownMatches(p, f) = \\ &\qquad \{(p, m, f) \in FriendMatches(p, f) : m \in Likes(f)\} \end{aligned} \tag{3}$$

$$knows(p, f) = \frac{|KnownMatches(p, f)|}{|FriendMatches(p, f)|} \tag{4}$$

2.1.2 Unknown Matches → Recommendations

Even when we have no information about f's interest in m ($m \notin Likes(f)$) this does not mean it is a poor match; $Likes(f)$ is not exhaustive. And the fact that p assigns the match suggests that p believes that f will be interested in m, establishing m as a plausible, novel recommendation candidate for f. If other players also match m with f then this strengthens the possible interest of f in m. Equation 5 captures this as an interest score based on the number of players who have matched some unknown m with f, and their tie strength with f.

$$interest(m, f) = \sum_{\forall p: match(p,m,f) \wedge m \notin Likes(f)} knows(p, f) \tag{5}$$

2.2 Game Mechanics

The game is implemented as a Facebook app to gain access to the social graph and likes of players; this way friends can be chosen and (known) matches can be verified. The game has been developed in PHP and is responsible for storing all player data, friendship links, movie data, and game sessions. The game uses the Rotten Tomatoes API to collect additional movie data (poster graphics, trailers etc.) for the purpose of enhancing gameplay. On the client-side the front-end is written in Javascript using jQuery and some task specific libraries for animation and image manipulation.

Game mechanics are simple and enjoyable. At the start of the game the player p is presented with a subset of friends; currently 5 friends are picked who have at least 10 movie likes. Each game comprises 18 movies which are chosen from a mixture of the friends' profiles/likes and the Rotten Tomatoes most popular movie list. This ensures a mix of known movies, with which to validate matches and infer tie strength, plus unknown movies to generate novel recommendation candidates.

During the game a graphic of a movie's poster floats across the screen following a particular trajectory and speed. As the game continues the screen becomes more cluttered and the trajectory of movies more erratic thereby making the matching process more frenetic and challenging. And when a player matches a movie by dragging its icon over a friend's avatar, the player is rewarded by an audio and graphical flourish and a score.

Scoring is more challenging than might first appear. It seems intuitive to award a score if the player matches a known (liked) movie with a friend. But if they match an unknown movie should they miss out, given how these matches are potentially valuable, novel recommendation candidates? If we only score known matches then players may avoid making more novel matches. And what if a novel match has been made often during other games? How should it effect the scoring?

In the end we chose the scoring metric shown in Equation 6 which combines scores for known and unknown matches; we set $\alpha = 0.5$ in this work but this could be adjusted to give more or less weight to each match type. This returns a score between up to 10 for each match. For example, if p matches m with f, such that $m \in Likes(f)$, and 3 other players have made the same match before, then p will receive a score 6.25. If on the other hand $m \notin \mathit{Likes}(f)$ and p is the first to make such a match then p receives a score of 5. There are no doubt many possible variations on this scoring metric that could (and should) be tested in the future.

$$score(p,m,f) = \\ 10 \bullet \left(\alpha \bullet 1[m \in \mathit{Likes}(f)] + \frac{1-\alpha}{1 + PastMatches(m,f)} \right) \qquad (6)$$

3. EVALUATION

To test the data generated by gameplay we ran a small two-part, live-user trial based on 27 mutual friends and friends-of-friends; a mixture of male and female undergraduates and postgraduates. Participants acted as both players and friends during gameplay. During each game a subset of 5 friends was chosen at random for a given p; the number of friends per participant varied from 5 to 12. For each participant we also had an initial set of movie likes from Facebook (on average, 27 movies per participant).

3.1 Methodology

The evaluation took place in two phases. Phase 1 focused on collecting data by asking participants to play the game a few times; in fact players played an average of 9 games each, suggesting many found it at least somewhat enjoyable. On average 11.69 matches were made per game. These data were then used in phase 2 to generate recommendations for each of the 27 users. We generated 6 movie recommendations per user from 3 different recommendation strategies and asked each participant to rate all 18 recommendations as either satisfactory or unsatisfactory; a simple binary rating per movie. We were careful to interleave the order of recommendations from each strategy to avoid any positional bias.

3.2 Recommendation Strategies

As to the recommendation strategies used: we distinguish each in terms of the candidate items they consider for recommendation and how these are ranked to produce a *top-6* for the target user u_t, as follows.

3.2.1 Crowdsourced Recommendations (CS)

Here we rely purely on gameplay data to generate and rank our recommendations. The candidates are those movies matched with u_t (but unknown to u_t) by player-friends of u_t during gameplay; see Equation 7. These candidates are then ranked in terms of $interest(m, u_t)$ as per Equation 5.

$$CS\_Candidates(u_t) = \\ \bigcup_{\forall p \in Friends(u_t)} \{m : match(p, m, u_t) \wedge m \notin \mathit{Likes}(u_t)\} \qquad (7)$$

Thus, movies that are often matched with u_t, by many players who know u_t's preferences well, will rank higher than movies less often matched with u_t by less familiar players.

3.2.2 Collaborative Filtering Recommendations (CF)

Next we implemented a version of collaborative filtering by choosing movies from the profiles (likes) of u_t's friends; see Equation 8. Collaborative filtering usually ranks recommendations based on the similarity between u_t and the friends/neighbours from where they originate; similarity is usually based on some form of ratings correlation. In our small trial we instead use the $knows(f, u_t)$ metric as a proxy for user similarity and rank the candidates according to $interest(m, u_t)$. In this way the CS and the CF techniques differ primarily in the source of the candidates (gameplay vs. profiles, respectively).

	Satisfaction	Diversity
CS	90%	21%
CF	79%	37%
CB	77%	50%

Table 1: Satisfaction and diversity results.

$$CF\_Candidates(u_t) = \\ \bigcup_{\forall f \in Friends(u_t)} \mathit{Likes}(f) - \mathit{Likes}(u_t) \qquad (8)$$

3.2.3 Content-based Recommendations (CB)

Finally, as a content-based strategy we used the Rotten Tomatoes API call, $movie\_similar(m)$, to obtain a set of 5 movies that are similar (based on meta-data) to each $m \in \mathit{Likes}(u_t)$. The candidate movies for u_t are the collection ($\bigoplus$)) of all movies returned by Rotten Tomatoes for each of the movies in $\mathit{Likes}(u_t)$.

$$CB\_Candidates(u_t) = \bigoplus_{\forall m \epsilon \mathit{Likes}(u_t)} movie\_similar(m) \qquad (9)$$

At recommendation time 6 content-based candidates are selected from these candidates at random. Since these collections may contain duplicates, if the same movies are returned by Rotten Tomatoes for different seeds, then this approach will tend to recommend movies that are more frequently represented in the candidate list, giving priority to movies that are considered similar to many of u_t's likes.

3.3 Results

Satisfaction results are shown in Figure 1 and indicate a benefit to the CS approach compared to CF or CB. 90% of users found the CS recommendations to be satisfactory compared to only 79% and 77% for CF and CB, respectively. This speaks to the quality of the recommendation data being created as part of the gameplay; remember the difference between CS and CF is only a matter of how the movies were sourced (gameplay vs. profiles).

It is also interesting to examine the diversity of recommendations made by the different approaches; diversity speaks to the ability of the recommender to offer the user more or less variety through its suggestions [13]. As a simple measure of diversity we compared the percentage of unique recommendations made by each of the 3 approaches in Figure 1. CS presents with lower levels of diversity than either CF or CB. Only 21% of the movies recommended by CS were unique compared to 37% and 50% for CF and CB.

One explanation for this is that the CS approach tends to skew towards popular movies because players recognise their posters more quickly and naturally gravitate towards these during gameplay. Moreover, because these movies are popular they may also be easier to match with friends leading to a greater likelihood that they will be liked, hence the improved satisfaction scores. The higher level of diversity using the CB approach on the other hand, can be partially attributed to the fact that it can draw recommendations from a larger pool of candidates.

It is a matter for future work to consider this relationship between satisfaction and diversity further. It may be possible to guide gameplay towards more novel items by manipulating the size and speed of items according to the item popularity. For example, perhaps unusual movies could be presented with a larger icon and/or a slower trajectory, making them more attractive gameplay targets; scoring could also be adapted with respect to inverse item popularity.

4. CONCLUSIONS

The purpose of this paper has been to explore the use of a GWAP to collect recommendation data as a side-effect of casual gameplay. We have implemented and tested a simple movie matching game and described the different types of recommendation data that we can collect from its gameplay. We have shown how this data can be useful in a recommendation context as part of a live-user trial. And, it should be noted that, as a practical matter, the approach is not limited to movie preferences because similar ideas could be used for many other types of items.

This work is of course limited in many ways. The system is a working prototype and the small scale of our evaluation offers little more than a proof-of-concept for what we are trying to achieve. Nevertheless we believe it serves as a first-step to highlight the potential for new ways to think about recommendation data and recommender systems while contributing to a growing interest in the role of crowdsourcing in recommender systems research.

5. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005.

[2] G. K. Christian Desrosiers. A Comprehensive Survey of Neighborhood-based Recommendation Methods. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 107–144. 2011.

[3] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, and F. Players. Predicting Protein Structures with a Multiplayer Online Game. *Nature*, 466(7307):756–760, 2010.

[4] A. Felfernig, M. Jeran, M. Stettinger, T. Absenger, T. Gruber, S. Haas, E. Kirchengast, M. Schwarz, L. Skofitsch, and T. Ulz. Human computation based acquisition of financial service advisory practices. In *Proceedings of the 1st International Workshop on Personalization & Recommender Systems in Financial Services, Graz, Austria, April 16, 2015.*, pages 27–34, 2015.

[5] R. Gligorov, M. Hildebrand, J. van Ossenbruggen, G. Schreiber, and L. Aroyo. On the Role of User-Generated Metadata in Audio Visual Collections. *Proceedings of the 6th International Conference on Knowledge Capture (K-CAP '11)*, pages 145–151, June 2011.

[6] J. Golbeck. In K. Stølen, W. H. Winsborough, F. Martinelli, and F. Massacci, editors, *Proceedings of the 4th International Conference on Trust Management (iTrust'06)*, pages 93–104, Pisa, Italy, May. Springer Berlin Heidelberg.

[7] S. Hacker and L. von Ahn. Matchin: eliciting user preferences with an online game. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009*, pages 1207–1216, 2009.

[8] D. Kelly and J. Teevan. Implicit Feedback for Inferring User Preference: A Bibliography. *ACM SIGIR Forum*, 37(2):18 – 28, 2003.

[9] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, Aug. 2009.

[10] E. Law and L. von Ahn. Input-Agreement: A New Mechanism for Collecting Data using Human Computation Games. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems - CHI 09*, pages 1197–1206, Boston, Massachusetts, USA, Apr. 2009. ACM Press.

[11] P. Organisciak, J. Teevan, S. Dumais, R. C. Miller, and A. T. Kalai. A Crowd of Your Own: Crowdsourcing for On-Demand Personalization. *Proceedings of the 2nd AAAI Conference on Human Computation and Crowdsourcing (HCOMP-2014)*, Nov. 2014.

[12] A. Salvador, A. Carlier, X. Giro-i Nieto, O. Marques, and V. Charvillat. Crowdsourced Object Segmentation with a Game. *Proceedings of the 2nd ACM International Workshop on Crowdsourcing for Multimedia - CrowdMM '13*, pages 15–20, Oct. 2013.

[13] B. Smyth and P. McClave. Similarity vs. diversity. In *Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, BC, Canada, July 30 - August 2, 2001, Proceedings*, pages 347–361, 2001.

[14] R. van Zwol, L. Garcia, G. Ramirez, B. Sigurbjornsson, and M. Labad. Video Tag Game. In H. Jinpeng, R. Chen, H.-W. Hon, Y. Liu, W.-Y. Ma, A. Tomkins, and X. Zhang, editors, *Proceedings of the 17th International World Wide Web Conference (WWW Developer Track)*, Beijing, China, Apr. 2008. ACM Press.

[15] L. von Ahn and L. Dabbish. Labeling Images with a Computer Game. *Proceedings of The ACM Conference on Human Factors in Computing Systems*, pages 319 – 326, Apr. 2004.

[16] G. Walsh and J. Golbeck. Curator: a Game with a Purpose for Collection Recommendation. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems - CHI '10*, pages 2079–2082, Atlanta, Georgia, USA, Apr. 2010. ACM Press.