The definitive version was published in Proceedings of the fourth ACM conference on Recommender systems 2010, Barcelona, Spain: September 26 - 30, 2010 http://doi.acm.org/10.1145/1864708.1864773

On the Real-Time Web as a Source of Recommendation Knowledge

Sandra Garcia Esparza, Michael P. O'Mahony, Barry Smyth
CLARITY: Centre for Sensor Web Technologies
School of Computer Science and Informatics
University College Dublin, Ireland
{sandra.garcia-esparza, michael.omahony, barry.smyth}@ucd.ie

ABSTRACT

The so-called real-time web (RTW) is a web of opinions, comments, and personal viewpoints, often expressed in the form of short, 140-character text messages providing abbreviated and personalized commentary in real-time. Twitter is undoubtedly the king of the RTW. It boasts 100+ million users and generates in the region of 50m tweets per day. This RTW data is far from the structured data (ratings, product features, etc.) familiar to recommender systems research, but it is useful to consider its applicability to recommendation scenarios. In this short paper we describe an experiment to look at harnessing the real-time opinions of movie fans, expressed through the Twitter-like short textual reviews available on the Blippr service (www.blippr.com). In particular we describe how users and movies can be represented from the terms used in their associated reviews and describe a number of experiments to highlight the recommendation potential of this RTW data-source and approach.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

Recommender Systems, Real-Time Web, Micro-blogging

1. INTRODUCTION

Recommender systems have proven to be an important way for people to discover information, products and services that are relevant to their needs. They complement more conventional query-based search services by offering more proactive information discovery, often based on a profile of the user's short or long-term preferences. There are two basic approaches to recommender systems: *collaborative filtering* (CF) [6, 10] and *content-based* (CB) [4, 11] approaches. The former model generates recommendations for a target user drawing on the items that similar users have liked, where user interests are typically represented as item ratings, and user similarity is based on correlations between ratings histories. In contrast, CB approaches take advantage of content-based item representations (e.g. movies can be described by meta-data such as genre, cast etc.) to drive item-similarity. Generating recommendations for a target user becomes a matter of identifying items that are similar to items that the user has already liked (or purchased). In addition, researchers have combined CF and CB methods as the basis for various types of hybrid strategies [3].

Often neither user ratings nor item meta-data are readily available to drive either CF or CB recommendation. The purpose of this paper is to explore a third source of recommendation data, in the form of *user-generated content* (UGC) that relates to items, products, and services, to cope with this type of situation. UGC is inherently noisy but it is plentiful, and recently researchers have started to consider its utility in recommendation. For example, the role of tags in recommender systems has been examined in [9]. Further, researchers have started to leverage user-generated reviews as a way to recommend and filter products and services. For example, [1] look at the use of user-generated movie reviews from IMDb as part of a movie recommender system and similar ideas are discussed in [12]. Both cases look to mine review content as an additional source of recommendation knowledge, but they rely on the availability of detailed item reviews which may not always be the case.

Here, we are interested in the more challenging form of user-generated content that comes from micro-blogging services like Twitter. Readers will be familiar with Twitter's short text messages (*tweets*), which allow users to broadcast their opinions on life, the universe and everything to anyone who cares to listen. Already researchers and practitioners alike have begun to enthuse about the potential for this type of UGC to influence the marketing of products and services [5]. Our interests run deeper, and in this paper we explore whether these fragmented and noisy snippets of user opinions can be used more directly in recommendation.

Sometimes tweets carry important preference-like information, or even a product review; for example, one recent new iPad owner tweeted: *"Typing this on iPad. I love it. With wireless keyboard I could see this as my laptop."* This tweet clearly expresses a positive opinion on Apple's latest creation and this *micro-review* carries important recom-

RecSys2010, September 26–30, 2010, Barcelona, Spain.

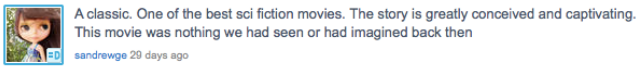

Figure 1: A Blippr review of the movie 'The Matrix'.

mendation information including specific information about features that work well. Given the volume of data that is generated by Twitter and other micro-blogging services, it makes sense to look at mining these data sources for recommendation tasks. To this end we consider two key questions: (1) Can RTW data be used as the basis for representing and recommending items, products and services? (2) How well does a recommender system based on RTW data perform in practice? In what follows, we describe experiments that are designed to shed light on these questions. Specifically, we develop a movie recommender system that is powered by Twitter-like movie-related comments and demonstrate strong recommendation performance on real user data.

2. HARVESTING THE RTW

In this paper we focus on a Twitter-like review service called Blippr (see www.blippr.com). Blippr allows registered users to express their views on a variety of product types including applications, music, movies, books and games. These reviews (or *blips*) are 160-character, text messages; further each user can indicate a rating on a 4-point rating scale (from "love it" to "hate it") and can also provide some tags. For instance, Figure 1 shows an example of a blip about the movie 'The Matrix'. For the purpose of this work we focus on Blippr data covering a total of 5,648 users, 5,934 movies and 18,750 tags. For clarity, we focus only on strong-positive blips (i.e. rated as "love it" by the user), which provides 19,619 individual blips. We removed stopwords, digits, special symbols (i.e. question marks, ampersands etc.), emoticons and letter repetitions when these appear more than 2 times (e.g. "goooood" remains "good"). We also exclude blips which contain non-English characters.

To begin with we consider whether this real-time web data can be used as a source of indexing and retrieval information. Consider a movie M_i which is associated with a set of blips and tags as per Equation 1. In turn, each blip is made up of a set of terms and so each movie can be represented as a set of terms (drawn from blips and tags) using a bag-of-words style approach [8] according to Equation 1.

$$M_i = \{b_1, ..., b_k\} \cup \{tag_1, ...tag_m\} = \{t_1, ..., t_n\} \quad (1)$$

In this way individual movies can be viewed as documents made up of the set of terms (words) contained in their associated blips and tags. We can create an index of these documents so that we can retrieve documents (that is movies) based on the terms that are present within blips and tags. Of course the information retrieval community provides a well understood set of tools and techniques for dealing with just this form of document representation and retrieval. For example, there are many ways to *weight* the terms that are associated with a given movie based on how representative or informative these terms are with respect to the movie in question. In this work we use the well known TFIDF approach [8] to term weighting.

Thus we can create a term-based index of movies $\mathbf{M}$, such that each entry $\mathbf{M_{ij}}$ encodes the importance of term t_j in movie M_i; see Equation 2. In this work we use Lucene[1] to provide this indexing, term-weighting and retrieval functionality; by using the term-vector for a target movie, M_T, as a query, we can retrieve a set of n similar movies, $M_1, ..., M_n$ from the index, ranked according to their similarity to M_T.

$$\mathbf{M_{ij}} = \text{TFIDF}(M_i, t_j, \mathbf{M}) \quad (2)$$

To test whether this is an effective way of representing movies, we focus on movies that have at least 5 blips and 5 tags; in total, there are 363 such movies. We generate 3 different movie indices: one based solely on blips ($\mathbf{M_b}$), one based solely on tags ($\mathbf{M_t}$), and one based on blips and tags ($\mathbf{M_{b+t}}$). For each target movie M_T, we treat 3 of its blips as 3 separate queries. We remove these blips from the indices and then allow Lucene to retrieve a ranked-list of movies in response to this query for the 3 separate indices. We repeat this approach for queries based on tags; that is, use 3 tags as queries, remove them from the index and perform retrieval. In this way we can evaluate the effectiveness of 6 retrieval systems based on different combinations of queries (blips versus tags) and indices (blips versus tags versus blips and tags). We generate result-lists ranging in size from 1 to 100 movies and in each case note the *hit ratio* – i.e. the percentage of times that M_T is returned by the retrieval system.

The results are shown in Figure 2(a), with each system represented by a single curve. The best performance is achieved when we use blips as queries over an index of blips and tags; for example, we retrieve the target movie about 36% of the time for result-lists of 20 movies. By comparison, tags as queries over an index of tags alone performs poorly, retrieving the target movie only 7% of the time for 20-movie result-lists. More generally, we see better performance when we use blips as queries compared to tags as queries; tags are just not expressive enough in the case of the Blippr data. Similarly, indexing using blips, or blips and tags, delivers better performance than indexing by tags alone. Figure 2(b) plots the position (rank) of the matching target movies (in the blips vs blips condition) as a function of the number of blips used to index the target movie. This shows superior retrieval performance in cases where there are more blips available for indexing. For example, when there are less than 50 blips per movie, then the position of the target movie varies between the top 5 and the top 80 movies. In contrast, once we have more than 50 blips per movie we see that when the target movie is found (which is approximately 80% of the time), it tends to be located among the top 20 results, and often in the top 10.

Clearly the above is a somewhat limited test of the utility of blips as a way to index items. Nevertheless, the results suggest that blips are a useful form of indexing information, at least sufficient to retrieve a given movie based on a subset of its blips. Moreover, even though blips are inherently noisy and unstructured, they are a superior source of indexing information than tags on their own. This bodes well for the use of real-time data as a source of recommendation knowledge, which we consider more directly in the next section.

[1] `http://lucene.apache.org/`

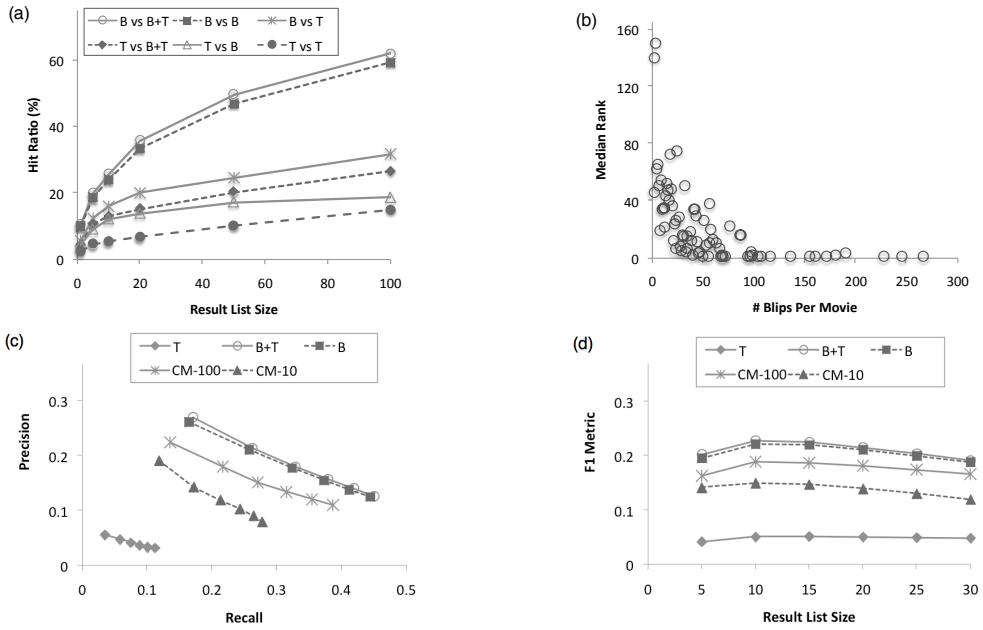

Figure 2: (a) Hit ratios for the 6 query-index combinations; (b) Median rank vs. movie index size; (c) Precision-recall and (d) F1 Metric for user-based (B vs. T vs. $B+T$) and community-based recommendation (CM-10 vs. CM-100).

3. RECOMMENDING MOVIES

To test the recommendation utility of real-time web data, we need to build a recommender system that is capable of recommending a set of movies for a given user. First we need to profile users. To do this, we adopt a similar approach to the manner in which we profiled movies. We treat each user as a document made up of their blips as per Equation 3; we could not obtain the sets of tags that individual users submitted from Blippr and so it is not possible to represent users by tags. As before, we can index the set of users using Lucene to produce a user index, $\mathbf{U}$, such that each entry $\mathbf{U_{ij}}$ encodes the importance of term t_j for user U_i, once again using Lucene's TFIDF scoring function as per Equation 4.

$$U_i = \{b_1, ..., b_k\} = \{t_1, ..., t_n\} \quad (3)$$

$$\mathbf{U_{ij}} = \text{TFIDF}(U_i, t_j, \mathbf{U}) \quad (4)$$

We now have two types of index for use in recommendation: an index of users, based on the terms in their blips, and an index of movies based on the terms in their blips (or in their tags or the combination of blips and tags). This suggests two different recommendation strategies. First, we can implement a *user-based* approach in which each user profile acts as a query against the movie index to produce a ranked-list of similar movies; see Figure 3. We test three variations on this approach, the first based on a movie index of blips (B), the second based on a movie index of tags (T), and the third based on a movie index of blips and tags ($B+T$).

Second, we implement a *community-based* approach based on collaborative filtering [10]. We identify a set of similar users, by using the target user profile as a query on the user index, and then rank the preferred movies of these similar users based on their frequency of occurrence in the similar user profiles; see Figure 4. We can adjust this algorithm by retrieving different numbers of similar users; in this case we compare the retrieval of 10 and 100 nearest neighbours.

Input: Target user U_T, user index **U**, movie index **M**, number of movies to retrieve n
Output: Top n movie recommendations

```
1.  UserBasedRecommendation (U_T, U, M, n)
2.  Begin
3.      query ← U.get(U_T)          // Return term vector for U_T in U
4.      recs ← M.retrieve(query)    // Retrieve ranked list of
                                    // movies from M based on query
5.      return recs.first(n)        // Return top n recommendations
6.  end
```

Figure 3: User-based recommendation.

To evaluate these algorithms on the Blippr movie dataset we build a user index from users who have between 5 and 20 blips (537 users) and we build a movie index from movies which have received at least 3 blips (1080 movies). We remove each user in turn from the user index to act as a target user, U_T. We also remove the blips from that user from the movie index. We know which movies this user has liked (the ones that they have rated with "love it") and we examine the overlap between these target movies and the movies generated by the user-based and community-based algorithms, calculating conventional precision and recall metrics for different recommendation-list sizes ranging from 5 to 30 movies.

The results are presented in Figure 2(c). There is a significant benefit for two of the user-based recommendation strategies (B and $B+T$) compared to the community-based approaches. As before, indexing movies using blips and tags shows a similar performance than an index based on blips alone, and much greater performance than an index based

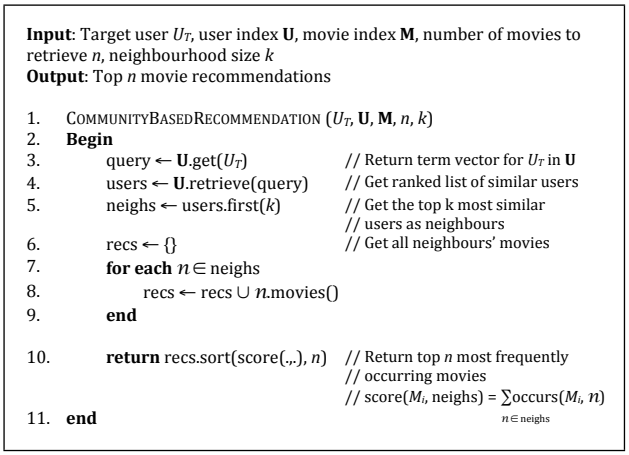

Figure 4: Community-based recommendation.

on tags alone (T). For example, for recommendation-lists of 5 movies we see that the user-based approach using blips and tags enjoys a precision score of 0.27, indicating that, on average, 1.35 of the 5 recommended movies are known to be liked by target users. Figure 2(c) also shows the community-based results when 10 (CM-10) and 100 (CM-100) similar users are selected as the basis for recommendation. There is clearly a benefit when it comes to drawing on a larger community of similar users, although our tests suggest that this does not extend beyond 100 users in practice, and neither approach is able to match the precision and recall scores of the user-based strategies.

Finally, it is worth highlighting the F1 scores in Figure 2(d), which are a harmonic mean of the precision and recall results. Obviously we see the same relative ordering of the 5 recommendation strategies, with the $B + T$ user-based indexing approach delivering almost a 20% increase in F1 score over the best performing community-based technique (CM-100). Interestingly, we also see that F1 is maximized for result-lists of size 10, indicating that the best balance of precision and recall is achieved for typical recommendation-list sizes.

4. CONCLUSIONS AND FUTURE WORK

In this paper we are interested in whether user-generated, micro-blogging messages, short and noisy as they are, have a role to play in recommender systems. We have described how to represent users and items based on micro-blogging reviews of movies and tested this technique using a number of recommendation strategies on live-user data. The results are promising. They suggest that micro-blogging messages can provide a useful recommendation signal, despite their short-form and inconsistent use of language. We have found that in this domain, adding user-generated tags to a blip-based index did not improve much the performance. However future work will study other types of content (e.g. metadata such as genre, cast etc.) and whether it can improve the performance of our UGC-based approach. Future studies will also compare our approach with classification techniques such as kNN, naïve Bayes, SVM etc.

This work is novel in its use of micro-blogging information for recommendation. Our approach is related to a growing body of research on the potential for UGC to inform recommendation [1, 2, 7, 12]. This related research focuses mainly on more conventional, long-form user reviews, whereas the work presented in this paper focuses on the more challenging micro-blogging messages.

5. ACKNOWLEDGEMENTS

Based on work supported by Science Foundation Ireland, Grant No. 07/CE/I1147.

6. REFERENCES

[1] S. Aciar, D. Zhang, S. Simoff, and J. Debenham. Recommender system based on consumer product reviews. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 719–723, Washington, DC, USA, 2006. IEEE Computer Society.

[2] S. Ahn and C.-K. Shi. Exploring movie recommendation system using cultural metadata. *Transactions on Edutainment II*, pages 119–134, 2009.

[3] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[4] S. Chelcea, G. Gallais, and B. Trousse. A personalized recommender system for travel information. In *UbiMob '04: Proceedings of the 1st French-speaking conference on Mobility and ubiquity computing*, pages 143–150, New York, NY, USA, 2004. ACM.

[5] B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Micro-blogging as online word of mouth branding. *CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, page 3859, 2009.

[6] 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*, 40(3):77–87, 1997.

[7] O. Phelan, K. McCarthy, and B. Smyth. Using twitter to recommend real-time topical news. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 385–388, New York, NY, USA, 2009. ACM.

[8] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[9] S. Sen, J. Vig, and J. Riedl. Tagommenders: connecting users to items through tags. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 671–680, New York, NY, USA, 2009. ACM.

[10] U. Shardanand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[11] B. Smyth and P. Cotter. A personalised TV listings service for the digital TV age. *Knowl.-Based Syst.*, 13(2-3):53–59, 2000.

[12] R. T. A. Wietsma and F. Ricci. Product reviews in mobile decision aid systems. In *PERMID*, pages 15–18, Munich, Germany, 2005.