Effective Product Recommendation using the Real-Time Web

Sandra Garcia Esparza, Michael P. O'Mahony and Barry Smyth

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 highly personalized commentary in real-time. Today, Twitter is undoubtedly the king of the RTW. It boasts 190 million users and generates in the region of 65m tweets per day[1]. This RTW data is far from the structured data (movie ratings, product features, etc.) that is familiar to recommender systems research but it is useful to consider its applicability to recommendation scenarios. In this paper we consider harnessing the real-time opinions of users, expressed through the Twitter-like short textual reviews available on the Blippr service (www.blippr.com). In particular we describe how users and products can be represented from the terms used in their associated reviews and describe experiments to highlight the recommendation potential of this RTW data-source and approach.

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. Recommender systems complement the more conventional query-based search services by offering more proactive information discovery, often based on a profile of users' short- or long-term preferences. It is useful to view many recommendation techniques as falling, broadly speaking, into one of two basic categories: *collaborative filtering* versus *content-based* approaches.

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

[1] http://techcrunch.com/2010/06/08/twitter-190-million-users/

In collaborative filtering approaches, items are selected for recommendation to some target user based on the items that similar users have liked in the past [24]. The key source of recommendation knowledge that collaborative filtering approaches use is the *ratings matrix*. This is a *user-item* matrix that captures the *interest* that a user U_i has in item I_j. Sometimes these interests are in the form of explicit ratings; for example, in MovieLens[2] users express their movie interests on the basis of a 1-5 rating scale. Other times these interests can be inferred from user actions; for example, Amazon's recommendations are based on user transaction histories and in this sense the purchasing of an item is viewed as a strongly positive rating. Very briefly, there are two flavours of collaborative filtering: (1) *user-based* techniques [18, 24] generate recommendations for a target user based on the items that similar users (that is, similarity among the rows of the ratings matrix) have liked in the past; (2) *item-based* approaches [21] generate recommendations based on the items that are similar to the items (that is, similarity among the columns of the ratings matrix) that the target user has liked in the past. Recent years has seen considerable research effort invested into this form of recommendation technique; in particular, focusing the manipulation of the core ratings matrix to better identify latent interests as a source of recommendation knowledge [9, 10].

Collaborative filtering approaches have been shown to work well when there is sufficient information to populate the ratings matrix, but very often this matrix is sparsely populated leading to poor coverage of the recommendation space and ultimately limiting recommendation effectiveness [2]. The alternative *content-based* approach to recommendation avoids the need for user ratings data, drawing instead on more richly detailed content representations of the items to be recommended [4]. For example, meta-data about a movie (genre, director, actors, etc.) can be used as the basis for item-level similarity assessment allowing content-based recommenders to rank items that are similar (content-wise) to the items that a target user is known to like (and perhaps dissimilar to the items that the target user is known to dislike). Content-based approaches have been used in a variety of recommendation applications including TV, e-commerce and travel [6, 22, 25]. In addition, researchers have looked at the potential to combine collaborative filtering and content based approaches as the basis for *hybrid* recommendation strategies [5]. A key challenge, however, relating to content-based systems is the overhead involved in obtaining the meta-data required to represent items; indeed, for some domains (e.g. jokes, works of art etc.), representing items effectively with such data can be problematic.

There is, however, a third source of recommendation data that can be considered. Most readers will be familiar with Twitter's short-form text messages (*tweets*), that allow users to broadcast their opinions on life, the universe and everything to just about anyone who cares to listen. Sometimes these messages carry important preference-like information or even a product review; for example, one recent new iPad owner posted: *"Typing this tweet on iPad. I love it. With wireless keyboard I could see this as my laptop replacement."* This tweet is clearly expressing a positive opinion on Apple's latest creation. Moreover, this type of 'review' carries some im-

[2] `http://www.grouplens.org`

portant recommendation information and not just simple sentiment, but also specific information about certain features (in this case, the wireless keyboard). Already researchers and practitioners alike have begun to enthuse about the potential for this type of user-generated content to influence the marketing of products and services [8]. 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. To this end we consider two important questions: (1) Can RTW data be used as the basis for representing, indexing, and recommending items, products and services? (2) How well does a recommender system based on RTW data perform relative to traditional approaches? In what follows we describe experiments that are designed to shed light on these important questions. Specifically, we develop a product recommender system that is powered by Twitter-like product-related comments and show that it has the potential to outperform a comparable collaborative filtering approach.

The paper is organized as follows. In Section 2, we describe related work that has been carried out on sentiment analysis and opinion mining of user-generated content. A description of the Blippr service[3], which we use as our test domain, is presented in Section 3. Our recommender approach, based on RTW data, is detailed in Section 4 and the results of an empirical evaluation of the approach are given in Section 5. Finally, we present concluding remarks in Section 6.

2 Related Work

In past years, user opinions in the form of reviews, comments, blogs and microblogs have been used by researchers for different purposes. One of the areas which has captured the interest of researchers is the application of sentiment analysis techniques to these opinions. In addition, user-generated content has also served as an additional source of knowledge for recommender systems. Here, we provide an overview of some of the work that has been carried out in this regard.

Sentiment analysis [26] encompasses areas such as subjectivity classification [28], opinion summarization [7] and review rating prediction [30]. Traditional text classification techniques based on machine learning have been applied in sentiment classification and indeed have proven their efficiency in many occasions [12, 14]. However, in [17] it is demonstrated how these models are topic-dependant, domain-dependant and temporally-dependant. Moreover, they suggest that relying on the emoticons present in the text — and using those texts as training data — can be a way of reducing the previous dependencies.

Lately, sentiment analysis techniques have also been applied to short texts like micro-blog messages. In [13], the authors present different machine learning techniques to classify Twitter messages as positive, negative or neutral. In order to do so, they create two classifiers: a neutral-sentiment classifier and a polarity (negative or positive) classifier. Extracting product features from reviews and identify-

[3] `http://www.blippr.com`

ing opinions associated with these features has also been studied in [16]. Our approach, however, is not aimed at employing sentiment analysis or opinion mining techniques; instead, we are interested in using user-generated content to provide better recommendations than traditional recommender systems.

Indeed, researchers have recently begun to consider the utility of such content as an additional source of recommendation data. For example, the role of tags in recommender systems has been examined in [23]. Further, researchers have started to leverage user-generated reviews as a way to recommend and filter products and services. For example, in [11, 15] the number of ratings in a collaborative filtering system is increased by inferring new ratings from user reviews using sentiment analysis techniques. Both works are evaluated on movie datasets (the former on Netflix and Flixster and the latter on IMDb).

In [29], another example is presented where a recommender system avails of user-generated content. They propose a hybrid collaborative filtering and content-based approach to recommend hotels and attractions, where the collaborative filtering component benefits from user-generated reviews. Moreover, they also comment on the advantages of using user-generated content for recommender systems; such as, for example, providing a better rationale for recommended products and increasing user trust in the system. Similar ideas are presented in [1], which look at using user-generated movie reviews from IMDb in combination with movie metadata (e.g. keywords, genres, plot outlines and synopses) as input for a movie recommender system. Their results show that user reviews provide the best source of information for movie recommendations, followed by movie genre data.

The approach proposed in this paper expands on the above work. In particular, our approach to product recommendation involves representing users and products based on the terms used in associated reviews, from which recommendations are subsequently made. In addition, we focus on short reviews from micro-blogging services as opposed to the longer-form product reviews that have typically been considered in previous work. In the next section, we describe the Blippr service, from where the micro-review data that is employed in our approach is sourced.

3 The Blippr Service

In this paper we focus on a Twitter-like review service called Blippr. This service allows registered users to review products from five different categories: *applications*, *music*, *movies*, *books* and *games*. These reviews (or *blips*) are in the form of 160-character text messages, and users must also supply an accompanying rating on a 4-point rating scale: *love it*, *like it*, *dislike it* or *hate it*. For instance, Figure 1 shows a screenshot of the Blippr interface when a user wants to add a new blip about the movie *The Matrix*. The user must add a review and a rating. In addition, the website shows past reviews for this movie from other users and their associated ratings.

Besides adding blips, users can also add tags to products. However, in order to avoid user abuse, Blippr currently does not allow users to tag popular products nor

Fig. 1 Adding a blip about a movie on the Blippr service.

to see which users added particular tags. Blippr also provides users with recommendations for the different product types, although precise details on the recommendation algorithm employed have not been published. Further, Blippr users can follow friends in a Twitter-like fashion and share their reviews with them. Finally, users can also post their blips to other services like Twitter or Buzz.

The Blippr service provides us with a useful source of real-time web data, which facilitates an analysis of the performance of recommendation algorithms across a range of product types. In the next section, we describe our recommendation techniques in detail and show how the micro-blogging activity of users can be harnessed to deliver effective product recommendations.

4 Product Recommendation using RTW Data

A key issue with collaborative and content-based recommenders is that oftentimes neither user ratings nor item meta-data are available to effectively drive either approach. In this paper, we explore a third source of recommendation data — namely, user-generated content relating to products and services — to deal with such situations. While user-generated content is inherently noisy, it is plentiful and here we describe an approach which uses this data in order to recommend products to users.

4.1 Index Creation

Our approach involves the creation of two indices, representing users and products, from which product recommendations are made to users. In this section, we consider how real-time web data can be used as a source of indexing information.

Product Index. We create this index as follows. Consider a product P_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 product can be represented as a set of terms (drawn from blips and tags) using a bag-of-words style approach [19] according to Equation 1.

$$P_i = \{b_1, ..., b_k\} \cup \{tag_1, ...tag_m\} = \{t_1, ..., t_n\} \tag{1}$$

In this way individual products 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 products) based on the terms that are present in their blips and tags. The information retrieval community provides a well understood set of 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 product based on how representative or informative these terms are with respect to the product in question. Here we use the well known TFIDF approach [19] to term weighting (Equation 2). Briefly, the weight of a term t_j in a product P_i, with respect to some collection of products $\mathbf{P}$, is proportional to the frequency of occurrence of t_j in P_i (denoted by n_{t_j,P_i}), but inversely proportional to the frequency of occurrence of t_j in $\mathbf{P}$ overall, thus giving preference to terms that help to discriminate P_i from the other products in the collection.

$$\text{TFIDF}(P_i, t_j, \mathbf{P}) = \frac{n_{t_j,P_i}}{\sum_{t_k \in P_i} n_{t_k,P_i}} \times \log\left(\frac{|\mathbf{P}|}{|\{P_k \in \mathbf{P} : t_j \in P_k\}|}\right) \tag{2}$$

Thus we can create a term-based index of products $\mathbf{P}$, such that each entry $\mathbf{P_{ij}}$ encodes the importance of term t_j in product P_i; see Equation 3. In this work we use Lucene[4] to provide this indexing and term-weighting functionality.

$$\mathbf{P_{ij}} = \text{TFIDF}(P_i, t_j, \mathbf{P}) \tag{3}$$

User Index. We use a similar approach to that above to create the user index. Specifically, we treat each user as a document made up of their blips (Equation 4); since we could not obtain the tags submitted by individual users from Blippr, it is not possible to represent users by tags. As before, we 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 5.

$$U_i = \{b_1, ..., b_k\} = \{t_1, ..., t_n\} \tag{4}$$

$$\mathbf{U_{ij}} = \text{TFIDF}(U_i, t_j, \mathbf{U}) \tag{5}$$

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

```
Input: Target user U_T, user index U, product index P, number of
products to retrieve n
Output: Top n product recommendations

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

Fig. 2 User-based recommendation algorithm.

4.2 Recommending Products

In the above, we have described how two types of index for use in recommendation are created: an index of users, based on the terms in their blips, and an index of products, based on the terms in their blips (or in their tags or the combination of blips and tags). This suggests the following recommendation strategies. First, we can implement a *user-based* approach in which the *target user's* profile acts as a query against the product index to produce a ranked-list of similar products[5]; see Figure 2. We consider three variations on this approach, the first based on a product index of blips (B), the second based on a product index of tags (T), and the third based on a product index of blips and tags ($B+T$).

In addition, to provide a benchmark for the above approach, we implement a *community-based* approach based on collaborative filtering ideas [24]. 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 products of these similar users based on their frequency of occurrence in the similar user profiles; see Figure 3. We can adjust this algorithm by retrieving different numbers of similar users; in Section 5 we compare the retrieval performance provided by using 10 and 100 nearest neighbours.

5 Evaluation

We now evaluate the recommendation performance provided by the RTW-based algorithms described above. We begin by describing the datasets used in our evaluation and the metrics that we employ to measure performance.

[5] The target user's blips are first removed from the product index to ensure that no bias is introduced into the process.

Input: Target user U_T, user index **U**, product index **P**, number of products to retrieve n, neighbourhood size k
Output: Top n movie recommendations

```
1.  CommunityBasedRecommendation (U_T, U, P, n, k)
2.  Begin
3.      query ← U.get(U_T)              // Return term vector for U_T in U
4.      users ← U.retrieve(query)       // Get ranked list of similar users
5.      neighs ← users.first(k)         // Get the top k most similar
                                        // users as neighbours
6.      recs ← {}                       // Get all neighbours' products
7.      for each n ∈ neighs
8.          recs ← recs ∪ n.products()
9.      end
10.     return recs.sort(score(.,.), n) // Return top n most frequently
                                        // occurring products
                                        // score(P_i, neighs) = ∑_{n∈neighs} occurs(P_i, n)
11. end
```

Fig. 3 Community-based recommendation algorithm.

5.1 Datasets

Our experiments use Blippr data relating to 4 different product types: *movies*, *books*, *applications* (apps) and *games*. As previously mentioned, Blippr facilitates feedback on items from 5 product types; in our work, we do not consider *music* products due to the small number of blips of this product type. For clarity, we focus on strong-positive blips only (i.e. where users have expressed the highest sentiment toward items). We collected data from the website using the Blippr API in April 2010, capturing blips written before that date (other data had to be scraped from the website due to the limitations of the API). We performed some preprocessing on the extracted blips such as removing stopwords, special symbols (?, &,*, etc.), digits and multiple repetitions of characters in words (e.g. we reduce *goooood* to *good*). Then we consider only blips that are written in English. For our experiments, we have selected those items that have received at least 3 blips and those users that have authored between 5 and 20 blips, inclusive. Dataset statistics are shown in Table 1.

Table 1 Statistics showing the number of items, tags and users present in each dataset.

Measure	Movies	Apps	Books	Games
# Items	1,080	268	313	277
# Users	542	373	120	164
# Blips	15,121	10,910	3,003	3,472
# Distinct Tags	1,543	817	649	165
Total # Tags	8,444	1,672	2,236	368

5.2 Metrics

We use *precision* and *recall*, which have been widely used in the field of information retrieval, to evaluate recommendation accuracy. These metrics have been adapted to evaluate the accuracy of a set of recommended items [20] and are defined as follows:

$$\text{Precision} = \frac{|T \cap R|}{|R|}, \qquad \text{Recall} = \frac{|T \cap R|}{|T|}, \tag{6}$$

where T is the test set and R is the recommended set of items for each user, respectively. Here, the test set for each user is given by the set of items that the user has blipped about (i.e. strong-positive blips of the user).

Precision and recall are often conflicting properties. For example, increasing the recommendation set size is likely to improve recall, but reduce precision. To resolve this conflict, we use the *F1 metric*, which is the harmonic mean of precision and recall [20, 27]. It is given by:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{7}$$

We also evaluate recommendation *coverage*, which measures the number of products for which a recommender is capable of making recommendations (as a percentage of the total number of products in the system). Clearly, the ability to make recommendations for as many products as possible is a desirable system property.

5.3 Recommendation Results

To evaluate our recommendation algorithms, we first create separate product and user indices for each of the 4 datasets according to the approach described in Section 4. For each dataset, we consider each user in turn from the user index to act as a target user, U_T, as per Section 4.2 and compute precision, recall and F1 metric scores for different recommendation-list sizes ranging from 5 to 30 movies.

Precision and recall results are presented in Figures 4–7 (left) for the *movies*, *applications*, *books* and *games* datasets, respectively. For all datasets, there is a clear benefit for two of the user-based recommendation strategies (B and $B+T$) compared to the community-based approaches. Indexing products using blips and tags, however, does not provide improved recommendation performance over an index based on blips alone; adding tags to the blip-based index achieves little or no effect. For example, in the case of the *books* dataset using recommendation lists of size 5, we see that both user-based approaches enjoy a precision score of approximately 0.34, indicating that, on average, almost 2 of the 5 recommended books are known to be liked by target users.

In all cases, an index based on tags alone (T) provides the worst recommendation performance. We also carried out experiments adding meta-data (e.g. title of the

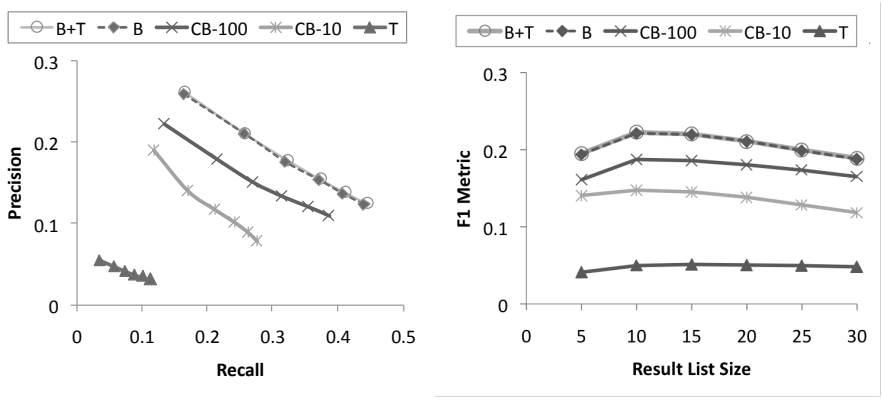

Fig. 4 *Movies* dataset: precision-recall (left) and F1 metric (right) for user-based (B vs. T vs. $B+T$) and community-based recommendation (CB-10 vs. CB-100).

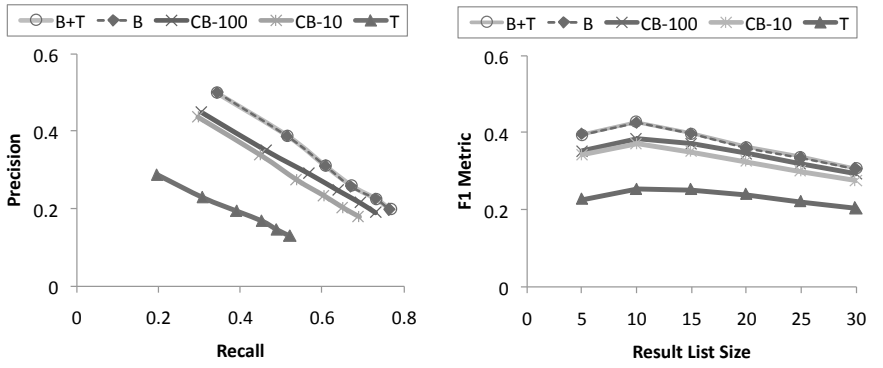

Fig. 5 *Applications* dataset: precision-recall (left) and F1 metric (right) for user-based (B vs. T vs. $B+T$) and community-based recommendation (CB-10 vs. CB-100).

product, genres, movie actors and directors, book authors and game platforms and developers) to the tag indices. Although some improvement in performance is seen using this approach for the tag index, overall the performance is still significantly worse compared to the other strategies and, in addition, the performance of indices based on blips and tags is not improved. We note, however, that tags and metadata may provide greater potential for recommendation in other domains, given the restrictions placed on adding tags to popular products on Blippr (see Section 3) and the relatively small numbers of tags present in our evaluation datasets.

Figures 4–7 (left) also show the community-based results when 10 (CB-10) and 100 (CB-100) similar users are selected as the basis for recommendation. For all except the *books* dataset, 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

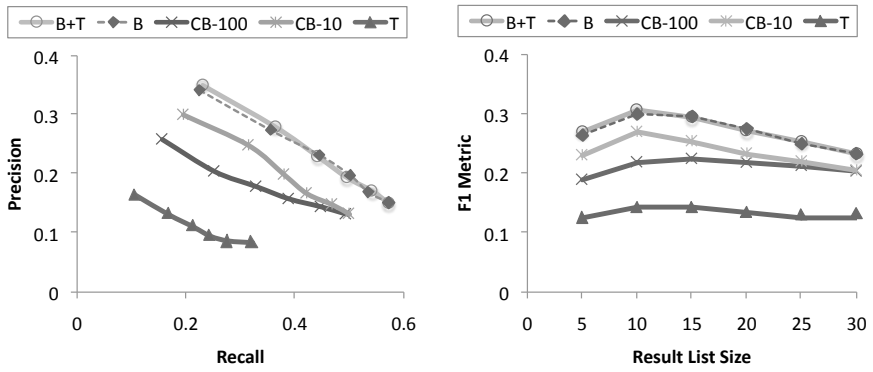

Fig. 6 *Books* dataset: precision-recall (left) and F1 metric (right) for user-based (B vs. T vs. $B+T$) and community-based recommendation (*CB*-10 vs. *CB*-100).

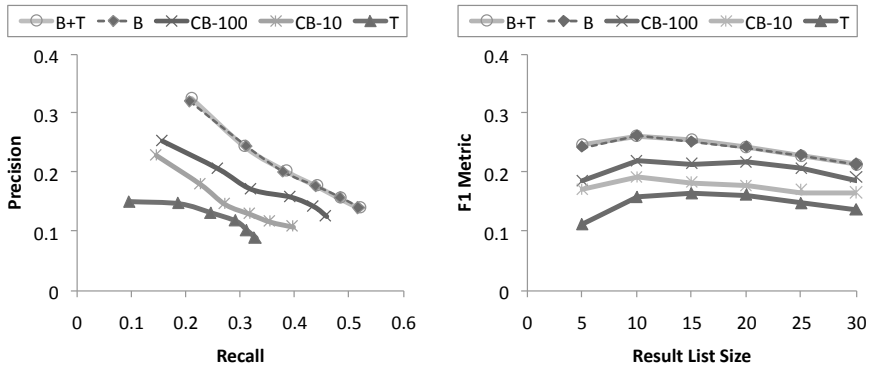

Fig. 7 *Games* dataset: precision-recall (left) and F1 metric (right) for user-based (B vs. T vs. $B+T$) and community-based recommendation (*CB*-10 vs. *CB*-100).

recall scores of the user-based strategies. The *books* dataset is the exception to this trend, where selecting 10 similar users achieves better performance than selecting 100 users (but did not outperform the blip-based index approach). This is likely due to the small number of users in this dataset; for example, the *books* dataset contains 120 users, compared to 542 users in the largest dataset (*movies*).

The F1 scores achieved by the 5 recommendation strategies are shown in Figures 4–7 (right). Obviously we see the same relative ordering of the different strategies as before with, for example, the user-based approach using a blip-based index delivering the best performance for all datasets. 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.

In Figure 8 (left), we compare the precision and recall performance provided by user-based recommendation using a blip-based index across the 4 datasets. It can

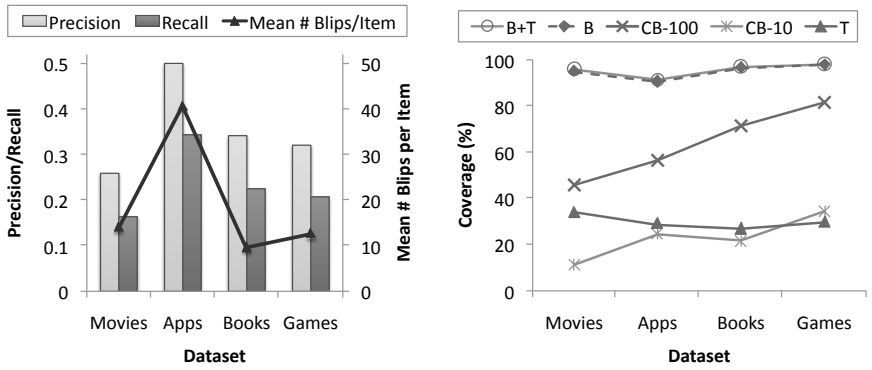

Fig. 8 Precision and recall (recommendation-list size = 5) provided by user-based recommendation using blip-based indices and mean number of blips per item for each dataset (left) and the coverage provided by the recommendation strategies for each dataset (right).

be seen that best performance is achieved for the *applications* dataset, with approximately similar trends observed for the other datasets. For example, precision and recall values of 0.50 and 0.34 are achieved for the *applications* dataset, respectively, compared to values of 0.34 and 0.23 for the *books* dataset (these values correspond to a recommendation list size of 5). Also shown in this figure is the mean number of blips per item for each dataset; it can be seen that these values correlate well with the precision (Pearson $r = 0.89$) and recall (Pearson $r = 0.90$) performance achieved for the datasets. This seems a reasonable finding, since it indicates that richer product indices (i.e. products which are described by a greater number of blips) lead to better recommendation performance. However, we note that the datasets used in our evaluation contain relatively small numbers of users, items and blips, and hence further analysis is required to make definitive conclusions in this regard.

Finally, we examine coverage performance in Figure 8 (right). The trends show that two of the user-based recommendation strategies (B and $B+T$) provide almost complete coverage for all datasets, well in excess of that given by both community-based approaches (even when using 100 nearest neighbours) and indexing by tags alone. These are very positive findings in respect of the utility of blips as a source of recommendation data, since they indicate that this approach is capable of providing significantly better coverage compared to the traditional community-based strategies, while being able to deliver more accurate recommendations as well.

6 Conclusions

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 4

product types 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 indices based on blips outperform a more traditional collaborative-filtering based approach in all the datasets evaluated.

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 user-generated content to inform recommendation [1, 3, 29]. 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. In future work, we will apply our approach to other domains like Twitter, which offers a rich source of user opinions on heterogeneous topics and products. In addition, we will expand our approach to recommend additional objects to users such as tags and other like-minded users and also consider the potential for cross-domain recommendation, where indices created using messages from one domain can be used to recommend products in other domains.

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

References

1. S. Aciar, D. Zhang, S. Simoff, and J. Debenham. Recommender system based on consumer product reviews. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 719–723, Washington, DC, USA, 2006. IEEE Computer Society.
2. 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, 2005.
3. S. Ahn and C.-K. Shi. Exploring movie recommendation system using cultural metadata. *Transactions on Edutainment II*, pages 119–134, 2009.
4. M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.
5. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
6. S. Chelcea, G. Gallais, and B. Trousse. A personalized recommender system for travel information. In *Proceedings of the 1st French-speaking conference on Mobility and ubiquity computing (UbiMob '04)*, pages 143–150, New York, NY, USA, 2004. ACM.
7. M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '04)*, pages 168–177, New York, NY, USA, 2004. ACM.
8. B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Micro-blogging as online word of mouth branding. *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems (CHI EA '09)*, pages 3859–3864, 2009.
9. Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 447–456, Paris, France, June 28–July 1 2009.
10. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

11. C. W.-k. Leung, S. C.-f. Chan, and F.-l. Chung. Integrating collaborative filtering and sentiment analysis: A rating inference approach. In *Proceedings of the ECAI 2006 Workshop on Recommender Systems*, pages 62–66, Riva del Garda, Italy, 2006.
12. T. Mullen and N. Collier. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*, 2004.
13. V. Pandey and C. Iyer. Sentiment analysis of microblogs. `http://www.stanford.edu/class/cs229/proj2009/PandeyIyer.pdf`, 2009. Accessed on: April 2010.
14. B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing (EMNLP '02)*, pages 79–86, Morristown, NJ, USA, 2002. Association for Computational Linguistics.
15. D. Poirier, I. Tellier, F. Franoise, and S. Julien. Toward text-based recommendations. In *Proceedings of the 9th international conference on Adaptivity, Personalization and Fusion of Heterogeneous Information (RIAO '10)*, Paris, France, 2010.
16. A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT '05)*, pages 339–346, 2005.
17. J. Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification, 2005.
18. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW 94)*, pages 175–186, Chapel Hill, North Carolina, USA, August 1994.
19. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.
20. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for ecommerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC '00)*, pages 158–167, Minneapolis, Minnesota, USA, October 17-20 2000. ACM.
21. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference (WWW '01)*, pages 285–295, Hong Kong, May 2001.
22. J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1-2):115–153, 2001.
23. S. Sen, J. Vig, and J. Riedl. Tagommenders: connecting users to items through tags. In *Proceedings of the 18th international conference on World wide web (WWW '09)*, pages 671–680, New York, NY, USA, 2009. ACM.
24. U. Shardanand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '95)*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.
25. B. Smyth and P. Cotter. A personalised TV listings service for the digital TV age. *Knowledge-Based Systems*, 13(2-3):53–59, 2000.
26. H. Tang, S. Tan, and X. Cheng. A survey on sentiment detection of reviews. *Expert Systems with Applications*, 36(7):10760–10773, 2009.
27. C. J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 1979.
28. J. Wiebe and E. Riloff. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of the 6th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing '05)*, pages 486–497, Mexico City, Mexico, 2005.
29. R. T. A. Wietsma and F. Ricci. Product reviews in mobile decision aid systems. In *Pervasive Mobile Interaction Devices (PERMID '05)*, pages 15–18, Munich, Germany, 2005.
30. Z. Zhang and B. Varadarajan. Utility scoring of product reviews. In *Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM '06)*, pages 51–57, New York, NY, USA, 2006. ACM.