<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Mining the Real-Time Web: A Novel Approach to Product Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authors(s)</strong></td>
<td>Garcia Esparza, Sandra, O'Mahony, Michael P., Smyth, Barry</td>
</tr>
<tr>
<td><strong>Publication date</strong></td>
<td>2012-05</td>
</tr>
<tr>
<td><strong>Publisher</strong></td>
<td>Elsevier</td>
</tr>
<tr>
<td><strong>Item record/more information</strong></td>
<td><a href="http://hdl.handle.net/10197/3746">http://hdl.handle.net/10197/3746</a></td>
</tr>
<tr>
<td><strong>Publisher's statement</strong></td>
<td>This is the author’s version of a work that was accepted for publication in Knowledge-Based Systems. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Knowledge-Based Systems (VOL 29, ISSUE May, (2012)) DOI:10.1016/j.knosys.2011.07.007</td>
</tr>
<tr>
<td><strong>Publisher's version (DOI)</strong></td>
<td>10.1016/j.knosys.2011.07.007</td>
</tr>
</tbody>
</table>

Downloaded 2024-06-05 09:03:17

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information
Mining the Real-Time Web: A Novel Approach to Product Recommendation

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.

Abstract

Real-time web (RTW) services such as Twitter allow users to express their opinions and interests, often expressed in the form of short text messages providing abbreviated and highly personalized commentary in real-time. Although this RTW data is far from the structured data (movie ratings, product features, etc.) that is familiar to recommender systems research, it can contain useful consumer reviews on products, services and brands. This paper describes how Twitter-like short-form messages can be leveraged as a source of indexing and retrieval information for product recommendation. In particular, we describe how users and products can be represented from the terms used in their associated reviews. An evaluation performed on four different product datasets from the Blippr service shows the potential of this type of recommendation knowledge, and the experiments show that our proposed approach outperforms a more traditional collaborative-filtering based approach.

Keywords: Real-Time Web, Micro-blogging, Recommender Systems, Information Retrieval

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-term or long-term preferences. These systems can now be found in many applications to provide users with a list of recommended items that they might like. For instance, the popular online retailer Amazon\(^1\), provides users with a list of product recommendations from different categories, while Last.fm\(^2\) does so for music and Netflix\(^3\) for movies.

Recommender systems typically fall into two basic categories: collaborative filtering (CF) \([9, 21, 18]\) versus content-based (CB) \([29, 33, 35]\) approaches. In collaborative filtering approaches, users are recommended items that users with similar interests have liked in the past \([46]\). The key source of recommendation knowledge used in collaborative filtering approaches is the ratings matrix. This is a user-item matrix that captures the particular interests that users have in items. Sometimes these interests are in the form of explicit ratings; for example, in MovieLens\(^4\) users express their movie interests on the basis of a 5-point 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. Traditionally, collaborative filtering methods belong to one of two groups: (1) user-based techniques \([46, 39]\), which 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 and (2) item-based approaches \([44]\), which generate recommendations based on 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 collaborative recommendation techniques; in particular, focusing the manipulation of the core ratings matrix to better identify latent interests as a source of recommendation knowledge \([26, 27]\).

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

\(^1\)http://www.amazon.com
\(^2\)http://www.last.fm
\(^3\)http://www.netflix.com
\(^4\)http://www.grouplens.org
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 [6]. For example, meta-data about a movie (genre, director, actors etc.) can be used as the basis for an 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). Sometimes the system’s users generate this meta-data, for example in social tagging systems, where users annotate resources using tags [13]. Applications of content-based recommenders include TV, e-commerce and travel applications [15, 45, 47]. A challenge 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. In addition, researchers have looked at the potential to combine collaborative filtering and content based approaches as the basis for hybrid recommendation strategies [12, 20]. However, there are cases where neither ratings nor meta-data are available in sufficient quantities to provide effective recommendation performance. For this reason, in this paper we consider a third source of recommendation data.

Recently micro-blog services have become very popular, especially since the release of Twitter in 2008, which generates huge volumes of data (currently Twitter users post 65M messages per day⁵). Micro-blogs allow users to broadcast their comments, opinions and interests on a wide variety of topics by way of short-form text messages. Sometimes these messages are related to products. For example, users tend to comment on new gadgets they have acquired or movies they have recently seen. For instance, Figure 1 shows an example of a tweet expressing a positive opinion on the movie ‘Inception’. These messages not only express an opinion about a product but sometimes they also contain specific information about certain features, such as the performance of a certain actor in a movie or the usability of a mobile phone’s interface.

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 [24]. Our interests run deeper, and in this

Figure 1: A tweet expressing a positive opinion on the movie ‘Inception’.

paper we explore whether these fragmented and noisy snippets of user opinions can be used more directly in recommendations. 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, which we use as our test domain, is presented in Section 3. Our recommender approach, based on RTW data, is described 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 recent years, user opinions in the form of reviews, comments, blogs and micro-blogs have been analyzed 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, including the auto extraction of product features from reviews, both well formed and unstructured. In addition, user-generated content (in long-form or short-form) 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.

http://www.blippr.com
A popular area of research lies in the application of sentiment analysis techniques to user-generated content [48]. Sentiment analysis encompasses different areas such as sentiment classification [10, 30, 32], which seeks to determine whether the semantic orientation of a piece of text is positive or negative (and sometimes, neutral). In [38] it was demonstrated that these sentiment classification models can be topic-dependant, domain-dependant and temporally-dependant and suggested that training with data which contains emoticons can make these models more independent. Another area within sentiment analysis is subjectivity classification [50], which classifies text as subjective (i.e. it contains author opinions) or objective (i.e. it contains factual information). Sometimes this task is applied before using a sentiment classification technique where the factual pieces of text could mislead the classifier.

Extracting product features from reviews and identifying opinions associated with these features has also been studied in [16, 22, 37]. Much of the initial work has focused on extracting features from electronic products such as cameras or MP3 players, where the set of product features is typically more restricted, hence representing a more tractable problem, compared to other domains such as movies or books. In more recent work [23], where feature extraction and opinion mining is performed on more complex (from a feature perspective) movie reviews, the authors first attempt to identify the set of key features that authors discuss by applying clustering techniques; a Latent Dirichlet Allocation approach was found to provide the best results. While some research [22, 37] applies feature extraction techniques, such as feature-based summarisation or point-wise mutual information in a domain-independent context, [16] argues that a domain-dependent approach is a valuable knowledge source, leading to a more precise feature set, and describe an approach based on a taxonomy of the domain product features.

Typically these feature and opinion mining approaches use Natural Language Processing (NLP) techniques which assume that the text is well-formed. Often, however, user-generated content is prone to noise in the form of, for example, spelling and grammatical errors. In [17], an approach is presented to extract features from noisy reviews by first identifying and correcting errors in the text before applying NLP techniques. Assisting users to write high quality reviews has also been studied in [11], where a case-based reasoning system is used to extract and suggest relevant product features to the review author from reviews that have been authored by others for similar products in the past. Identifying spam in product reviews is also an
important factor to improve the quality of reviews. This problem has been addressed in [25] where Amazon reviews are classified using a logistic regression model based on attributes relating to the review (for example, the volume of feedback received), the review author (for example, the number of times they were the first to review a product) and the reviewed product (for example, the sales rank of the product).

While the research and techniques described above have focused primarily on long-form review text, recent work has also considered the analysis of short-form reviews, such as micro-blog messages. For instance, in [31] Twitter messages are classified as positive, negative or neutral by creating two classifiers: a neutral-sentiment classifier and a polarity (negative or positive) classifier. Further [8] compare the effect of different attribute sets on sentiment classification for short-form and long-form reviews. Results show that while classification accuracy for long-form reviews can benefit from using more complex attribute sets (for example, bigrams and POS tagging), this is not the case for short-form reviews where simpler attributes based on unigrams alone were sufficient from a performance perspective. Spam detection for micro-blogs has also been addressed in [19], indicating that people are more susceptible to click on spam links contained in Twitter posts rather than those present in emails. Further, mining users’ interests and hot topics from micro-blog posts have also been investigated in recent research [4, 7].

Lately, researchers have started to leverage user-generated reviews as an additional source of recommendation data. For example, a methodology to build a recommender system which leverages user-generated content is described in [51]. Although an evaluation is not performed, they propose a hybrid of a collaborative filtering and a content-based approach to recommend hotels and attractions, where the collaborative filtering component utilises the review text to compute user similarities in place of traditional preference-based similarity computations. 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. One of the first attempts to build a recommender system based on user-generated review data is described in [1]. Here, an ontology is used to extract concepts from camera reviews and recommendations are provided based on users’ requests about a product; for example, “I would like to know if Sony361 is a good camera, specifically its interface and battery consumption”. In this case, the features interface and battery are identified, and for each of them a score is computed according to
the opinions (i.e. polarities) of other users and presented to the user.

Similar ideas are described in [3], which look at using user-generated movie reviews from IMDb in combination with movie meta-data (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. In addition, in [23, 36], the number of ratings in a collaborative filtering system is increased by inferring new ratings from user reviews using sentiment analysis techniques. While [36] generate ratings for Flixster reviews by extracting the overall sentiment expressed in the review, [23] extract features and their associated opinions from IMDb reviews and a rating is created by averaging the opinion polarities (i.e. positive or negative) across the various features. Both approaches achieve better performance when using the ratings inferred from reviews when compared to using ratings predicted by traditional collaborative filtering approaches. Further, extracting an overall preference rating [28, 53] or multiple ratings (reflecting multiple product aspects) [5] from reviews with the objective of providing users with summaries of reviewer sentiment on products has also been addressed.

The approach proposed in this paper expands on the above work. The key objective is to determine whether real-time web information in the form of short-textual reviews can be used as a useful source of recommendation knowledge. In particular, our approach to product recommendation involves representing users and products based on the terms used in their associated reviews, from which recommendations are subsequently made. Sentiment analysis or noise removal techniques have not been considered; an analysis of such techniques we leave to future work. In the next section, we describe the Blippr service, from which 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 2 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 and nor does it indicate 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 Facebook.

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 in sufficient quantity 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. Here, 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 (and tag) is made up of a set of terms and so each product can be represented as a set of terms using a bag-of-words style approach [42] according to Equation 1.

$$P_i = \{b_1, \ldots, b_k\} \cup \{tag_1, \ldots, tag_n\} = \{t_1, \ldots, t_n\}.$$  (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. One of the approaches to term weighting is term frequency–inverse document frequency (TF-IDF) [42], which is shown in Equation 2. Briefly, the weight of a term $t_j$ in a product $P_i$, with respect to some collection of products $P$, 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 $P$ overall, thus giving preference to terms that help to discriminate $P_i$ from the other products in the collection.

$$\text{TF-IDF}(P_i, t_j, P) = \frac{\text{tf}(t_j, P_i)}{\sum_{t_k \in P_i} \text{tf}(t_k, P_i)} \times \text{idf}(t_j, P),$$  (2)
where the term frequency, \( tf(t_j, P_i) \), and the inverse document frequency, \( idf(t_j, P) \), are given by Equations 3 and 4.

\[
\begin{align*}
  tf(t_j, P_i) &= n_{t_j, P_i}, \quad (3) \\
  idf(t_j, P) &= \log \left( \frac{|P|}{| \{ P_k \in P : t_j \in P_k \} |} \right). \quad (4)
\end{align*}
\]

Another popular term-weighting scheme is BM25, sometimes referred to as Okapi weighting [40]. This scheme also uses term frequency and inverse document frequency to weight terms. Using BM25, the importance of a term \( t_j \) in a product \( P_i \) is calculated as:

\[
BM25(P_i, t_j, P) = \sum_{k1} \frac{tf(t_j, P_i)}{k1 \times ((1 - b) + b \times \frac{|P|}{L(P_i)} + tf(t_j, P_i)) \times idf(t_j, P)},
\]

where \( L(P_i) \) is the length of the document in words and \( k1 \) and \( b \) are tuning constants. The constant \( k1 \) is used to control the influence of term-frequency in the equation (setting \( k1 = 0 \) removes the influence of term frequency). In TREC programme experiments [40], it was found that setting \( k1 = 2 \) provided good performance and it is typically used as the default value. The constant \( b \) controls the level of document length normalisation performed. It ranges in value from 0 to 1, where 0 means no normalisation and 1 is equivalent to a full normalisation. Typically this value is set to 0.75, which also was found helpful in TREC. Although we can use default values for \( k1 \) and \( b \), it is recommended that these values are set after systematic trials on the particular collection [41]. The inverse document frequency weight used in BM25 is usually a slight variation on that used in the above TF-IDF definition, and is given by:

\[
idf(t_j, P) = \log \left( \frac{|P| - \eta(t_j) + 0.5}{\eta(t_j) + 0.5} \right), \quad (6)
\]

where \( \eta(t_j) = | \{ P_k \in P : t_j \in P_k \} | \). Thus we can create a term-based index of products \( P \), such that each entry \( P_{ij} \) encodes the importance of term \( t_j \) in product \( P_i \), where term weights are calculated according to TF-IDF.
In this work we use Lucene\(^7\) which provides the TF-IDF term-weighting functionality. In addition, we use a BM25 extension for Lucene provided in [34].

\[ P_{ij} = \text{TF-IDF}(P_i, t_j, P) . \]  
\[ P_{ij} = \text{BM25}(P_i, t_j, P) . \] 

*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 as per Equation 9. (Note that we cannot represent users by tags given that Blippr does not reveal which tags added by individual users.) As before, we index the set of users using Lucene to produce a user index, \( U \), such that each entry \( U_{ij} \) encodes the importance of term \( t_j \) for user \( U_i \), once again using the TF-IDF or BM25 weighting functions as per Equations 10 and 11, respectively.

\[ U_i = \{b_1, ..., b_k\} = \{t_1, ..., t_n\} . \]  
\[ U_{ij} = \text{TF-IDF}(U_i, t_j, U) . \]  
\[ U_{ij} = \text{BM25}(U_i, t_j, U) . \] 

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 and an index of products, based on the terms in their associated blips (and tags in the case of products). 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 (the target user’s blips are first removed from the product index to ensure that no bias is introduced into the process); see Figure 3. We can consider different variations of this approach by using different weighting schemes (TF-IDF and

\(^7\text{http://lucene.apache.org}\)
BM25) to index and query the index. We can also apply stemming (i.e. reducing words to their stem, base or root form) to our data to improve the match between query and index terms.

In addition, to provide a benchmark for the above index-based approaches, we implement a community-based approach based on collaborative filtering ideas [46]. We identify a set of similar users (or neighbours), by using the target user profile as a query on the user index, and then rank the preferred products of these neighbours based on their frequency of occurrence in neighbour profiles; see Figure 4. We can adjust this algorithm by retrieving different numbers of neighbours; in our experiments performed in Section 5, we compare the retrieval performance provided by using 10 and 100 nearest neighbours.

![Community-based recommendation algorithm.](image)

**Figure 3: User-based recommendation algorithm.**

![Community-based recommendation algorithm.](image)

**Figure 4: Community-based recommendation algorithm.**

Input: Target user $U_T$, user index $U$, product index $P$, number of products to retrieve $n$

Output: Top $n$ product recommendations

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

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. **COMMUNITYBASEDRERECOMMENDATION** $(U_T, U, P, n, k)$
2. **Begin**
3. query $\leftarrow U.get(U_T)$ // Return term vector for $U_T$ in $U$
4. users $\leftarrow U.retrieve(query)$ // Get ranked list of similar users
5. neighs $\leftarrow users.first(k)$ // Get the top $k$ most similar users as neighbours
6. recs $\leftarrow ()$ // Get all neighbours' products
7. **for each** $n \in$ neighs
8. recs $\leftarrow$ recs $\cup$ neigh.products()
9. **end**
10. **return** recs.sort(score(.), n) // Return top $n$ most frequently occurring products
11. **end**
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.

5.1. 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 for this product type. For clarity, we focus on strong-positive blips only (i.e. where users have expressed the highest sentiment toward products). 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 gooood to good). We also consider only those blips that are written in the English language.

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 products [43] and are defined as follows:

\[
\text{Precision} = \frac{|T \cap R|}{|R|},
\]

\[
\text{Recall} = \frac{|T \cap R|}{|T|},
\]

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 products that the user has blipped about (since we only consider the strong-positive blips authored by users in our datasets).

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 \([49]\). It is given by:

\[
F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.
\]  

(14)

We also evaluate recommendation \emph{coverage}, which measures the number of products that a recommender is capable of making recommendations for (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. Indexing Results

In order to evaluate the utility of our approach as a means to represent products, we conducted initial experiments on one of the product indices (the \emph{movies} index). The objective of this experiment is to see if we can index and retrieve products represented by their blips and tags. In order to do so, we create three variations of the product index based on using: blips only \((B)\), tags only \((T)\) and both blips and tags \((B+T)\). For this experiment, TF-IDF is used as the weighting scheme. We focus on movies that have received at least 5 blips and 5 tags, which gives a total of 363 movies and 1,066 distinct tags. 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 these queries over the 3 index variations. 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 \emph{hit ratio} – i.e. the percentage of times that \(M_T\) is returned by the retrieval system.

The results are shown in Figure 5 (left), 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 \((B \text{ vs } B+T)\); for example, we retrieve the target movie about 36% of the time for result-lists of size 20 movies. Further, this performance is very close to using an index of blips alone, which indicates that tags do not contribute significantly to performance. Proof of this is the poor performance achieved when using tags as queries over an index of tags alone \((T \text{ vs } T)\), 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 5 (right) 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 example, when there are less than 50 blips per movie, the position of the target movie typically 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.

These results suggest that blips, albethey inherently noisy and unstructured, are a useful form of indexing information, at least sufficient to retrieve a given product based on a subset of its blips. 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. Given that tags did not prove to be useful for indexing and retrieving products in the movies dataset (or in any of the other datasets)\(^8\), in the following experiments products are represented by

\(^8\)This may be due, at least in part, to the restrictions placed on tagging popular movies
their blips only.

5.4. Recommendation Results

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 products for a given user. To evaluate our recommendation algorithms, we first need to create separate product and user indices for each of the 4 datasets according to the approach described in Section 4. For these 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. Statistics for all datasets are shown in Table 1.

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

<table>
<thead>
<tr>
<th></th>
<th>movies</th>
<th>apps</th>
<th>books</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td># Products</td>
<td>1,080</td>
<td>268</td>
<td>313</td>
<td>277</td>
</tr>
<tr>
<td># Users</td>
<td>542</td>
<td>373</td>
<td>120</td>
<td>164</td>
</tr>
<tr>
<td># Blips</td>
<td>15,121</td>
<td>10,910</td>
<td>3,003</td>
<td>3,472</td>
</tr>
<tr>
<td># Distinct Tags</td>
<td>1,543</td>
<td>817</td>
<td>649</td>
<td>165</td>
</tr>
<tr>
<td># Total Tags</td>
<td>8,444</td>
<td>1,672</td>
<td>2,236</td>
<td>368</td>
</tr>
</tbody>
</table>

Our main objective is to compare the performance of our user-based approach with that of the community-based benchmark (Section 4.2). In the case of the user-based approach, we are also interested in comparing the performance of two the popular weighting schemes: TF-IDF and BM25. Further, we want to determine if stemming has any effect on the performance of our recommender system. To do so, we compare a TF-IDF index without stemming (TF-IDF) to a TF-IDF index with stemming (TF-IDF+). For BM25, we performed tests to determine the optimal values for the tuning constants $k_1$ and $b$. We chose F1 corresponding to a result-list size of 5 (F1@5) as the reference metric to select the optimal values. From Table 2, it can be seen that the optimal values are different for all datasets. In particular, for the games dataset, the best F1@5 performance was achieved when $k1 = 0$, meaning that term weights are only influenced by inverse document frequency, with term frequencies playing no role (see Equation 5).

by the Blippr service. In other domains, tags may prove to be a more useful source of indexing information.
Table 2: Optimal BM25 parameters for each dataset.

<table>
<thead>
<tr>
<th></th>
<th>movies</th>
<th>apps</th>
<th>books</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>1</td>
<td>2</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>$b$</td>
<td>0.25</td>
<td>0.25</td>
<td>0.5</td>
<td>–</td>
</tr>
</tbody>
</table>

To perform the evaluation, 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 items. Precision and recall results are presented in Figures 6–9 (left) for the movies, applications, books and games datasets, respectively. For all datasets, there is a clear benefit for the user-based recommendation strategies compared to the community-based approaches. For example, in the case of the books dataset using recommendation lists of size 5, we see that the best user-based approach enjoys a precision score of approximately 0.44, indicating that, on average, more than 2 of the 5 recommended books are known to be liked by target users. Figures 6–9 (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 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).

For all datasets, TF-IDF with and without stemming provide similar results; with stemming applied, TF-IDF performs marginally better for most datasets. Similar trends are observed for BM25 (in the figures, the results without stemming are not shown). For the larger datasets (movies and apps), the performance provided by BM25 is very close to that of TF-IDF. In contrast, BM25 performance is seen to fall off for the smaller datasets (books and games), with for example, the community-based approach using 100 neighbours ($CB$-100) outperforming BM25 for the games dataset when rec-
ommendation list sizes of greater than 15 are considered.

The F1 scores achieved by the 5 recommendation strategies are shown in Figures 6–9 (right). Obviously we see the same relative ordering of the different strategies as before with the user-based approach employing TF-IDF with or without stemming delivering the best performance for all datasets. Interestingly, we also see that F1 is maximized for result-lists of size 10, followed closely by result-lists of size 5, indicating that the best balance of precision and recall is achieved for typical recommendation list sizes.

In Figure 10 (left), we compare the precision and recall provided by the user-based approach using TF-IDF across the 4 datasets. It can be seen that the best performance is achieved for the apps dataset, with approximately similar trends observed for the other datasets. For example, precision and recall values of 0.54 and 0.37 are achieved for the applications dataset, respectively, compared to values of 0.42 and 0.29 for the books dataset (these values correspond to recommendation lists of size 5). Also shown in this figure is the mean number of blips per product for each dataset; it can be seen that these values correlate well with the precision (Pearson $r = 0.84$) and recall (Pearson $r = 0.83$) performance achieved for the datasets. This seems a reasonable finding, since it indicates that richer product indices (i.e. products are described by a greater number of blips) lead to better recommendation performance. However, we note that the datasets used in our evaluation con-
Figure 7: Applications dataset: precision-recall (left) and F1 metric (right) for user-based (TF-IDF vs. TF-IDF+ vs. BM25) and community-based (CB-10 vs. CB-100) recommendation.

Figure 8: Books dataset: precision-recall (left) and F1 metric (right) for user-based (TF-IDF vs. TF-IDF+ vs. BM25) and community-based (CB-10 vs. CB-100) recommendation.
Figure 9: Games dataset: precision-recall (left) and F1 metric (right) for user-based (TF-IDF vs. TF-IDF+ vs. BM25) and community-based (CB-10 vs. CB-100) recommendation.

The Games dataset contains relatively small numbers of users, products and blips, and hence further analysis is required to make definitive conclusions in this regard.

Finally, we examine coverage performance in Figure 10 (right). Here we show the trends for the user-based recommendation strategy using TF-IDF and for the best performing community-based approach using 100 nearest neighbours (CB-100). It can be seen that the user-based approach provides almost complete coverage for all datasets, well in excess of that given by the community-based approach, particularly for the larger datasets (movies and apps). This is a very positive finding in respect of the utility of blips as a source of recommendation data, since it indicates that not only is this approach capable of providing significantly better coverage compared to the traditional community-based strategy, it is capable of delivering more accurate recommendations as well.

6. Conclusions

This paper investigates how user-generated micro-blogging messages can be used as a new source of recommendation knowledge. We have proposed an approach to represent users and products based on the terms in their associated reviews using techniques from the information retrieval community. We
have performed an evaluation on micro-blog reviews from 4 product datasets and the results are promising. First, they suggest that micro-blogging messages can provide a useful recommendation signal, despite their short-form and inconsistent use of language. Secondly, we have found that our approach outperforms a more traditional collaborative-filtering based approach in all the datasets evaluated, both in terms of accuracy and coverage. In future work, we will consider other performance metrics in our evaluations such as novelty [14] and diversity [52].

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 provide product recommendations [1, 3, 23]. This related research focuses mainly on more conventional, long-form user reviews, whereas the work presented here focuses on the more challenging micro-blogging messages. In future work, we will apply our approach to other domains such as Twitter which offers a rich source of user opinions and interests on heterogeneous topics and products. We will also focus on enriching user and item profiles by using sentiment analysis and feature extraction techniques. Further, we will expand our approach to consider the potential for cross-domain recommendation, where indices created using messages from one domain are used to recommend products from other domains.
7. Acknowledgements
Based on work supported by Science Foundation Ireland, Grant No. 07/CE/I1147.

References


