Title	Mining the Real-Time Web: A Novel Approach to Product Recommendation
Authors(s)	Garcia Esparza, Sandra, O'Mahony, Michael P., Smyth, Barry
Publication date	2012-05
Publication information	Garcia Esparza, Sandra, Michael P. O'Mahony, and Barry Smyth. "Mining the Real-Time Web: A Novel Approach to Product Recommendation." Elsevier, May 2012. https://doi.org/10.1016/j.knosys.2011.07.007.
Publisher	Elsevier
Item record/more information	http://hdl.handle.net/10197/3746
Publisher's statement	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
Publisher's version (DOI)	10.1016/j.knosys.2011.07.007

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

*Principal corresponding author

Email addresses: sandra.garcia-esparza@ucd.ie (Sandra Garcia Esparza), michael.omahony@ucd.ie (Michael P. O'Mahony), barry.smyth@ucd.ie (Barry Smyth)

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.

Recommendater 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 day[5]). 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

[5] `http://techcrunch.com/2010/06/08/twitter-190-million-users/`. Accessed: 1st December, 2010.

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[6] 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.

[6]`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'*.

Figure 2: Adding a blip for the movie *'The Matrix'* on the Blippr service.

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.

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

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. 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 $\mathbf{P}$, 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.

$$\text{TF-IDF}(P_i, t_j, \mathbf{P}) = \frac{\text{tf}(t_j, P_i)}{\sum_{t_k \in P_i} \text{tf}(t_k, P_i)} \times \text{idf}(t_j, \mathbf{P}) \ , \tag{2}$$

where the *term frequency*, $\mathrm{tf}(t_j, P_i)$, and the *inverse document frequency*, $\mathrm{idf}(t_j, \mathbf{P})$, are given by Equations 3 and 4.

$$\mathrm{tf}(t_j, P_i) = n_{t_j, P_i} \ . \tag{3}$$

$$\mathrm{idf}(t_j, \mathbf{P}) = \log\left(\frac{|\mathbf{P}|}{|\{P_k \in \mathbf{P} : t_j \in P_k\}|}\right) . \tag{4}$$

Another popular term-weighting scheme is BM25, sometimes referred to as *Okapi* weigthing [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:

$$\mathrm{BM25}(P_i, t_j, \mathbf{P}) = \sum \frac{\mathrm{tf}(t_j, P_i)}{k_1 \times ((1-b) + b \times \frac{|\mathbf{P}| \times L(P_i)}{\sum_{P_k \in \mathbf{P}} L(P_k)}) + \mathrm{tf}(t_j, P_i)} \times \mathrm{idf}(t_j, \mathbf{P}) \ , \tag{5}$$

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:

$$\mathrm{idf}(t_j, \mathbf{P}) = \log\left(\frac{|\mathbf{P}| - \eta(t_j) + 0.5}{\eta(t_j) + 0.5}\right) , \tag{6}$$

where $\eta(t_j) = |\{P_k \in \mathbf{P} : t_j \in P_k\}|$. 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, where term weights are calculated according to TF-IDF

(Equation 7) or BM25 (Equation 8). 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].

$$\mathbf{P_{ij}} = \text{TF-IDF}(P_i, t_j, \mathbf{P}) \ . \tag{7}$$

$$\mathbf{P_{ij}} = \text{BM25}(P_i, t_j, \mathbf{P}) \ . \tag{8}$$

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, $\mathbf{U}$, such that each entry $\mathbf{U_{ij}}$ 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.

$$U_i = \{b_1, ..., b_k\} = \{t_1, ..., t_n\} \ . \tag{9}$$

$$\mathbf{U_{ij}} = \text{TF-IDF}(U_i, t_j, \mathbf{U}) \ . \tag{10}$$

$$\mathbf{U_{ij}} = \text{BM25}(U_i, t_j, \mathbf{U}) \ . \tag{11}$$

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] 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

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

Figure 3: User-based recommendation algorithm.

```
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

CommunityBasedRecommendation (U_T, U, P, n, k)
Begin
    query ← U.get(U_T)            // Return term vector for U_T in U
    users ← U.retrieve(query)     // Get ranked list of similar users
    neighs ← users.first(k)       // Get the top k most similar
                                  // users as neighbours
    recs ← {}                     // Get all neighbours' products
    for each n ∈ neighs
        recs ← recs ∪ n.products()
    end

    return recs.sort(score(.,.), n)   // Return top n most frequently
                                      // occurring products
                                      // score(P_i, neighs) = ∑_{n ∈ neighs} occurs(P_i, n)
end
```

Figure 4: Community-based recommendation algorithm.

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.

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 *goooood* 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:

$$\text{Precision} = \frac{|T \cap R|}{|R|} \ , \tag{12}$$

$$\text{Recall} = \frac{|T \cap R|}{|T|} \ , \tag{13}$$

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:

$$\mathrm{F1} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \,. \tag{14}$$

We also evaluate recommendation *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 *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 *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 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 vs T), retrieving the target movie only 7% of the time for 20-movie

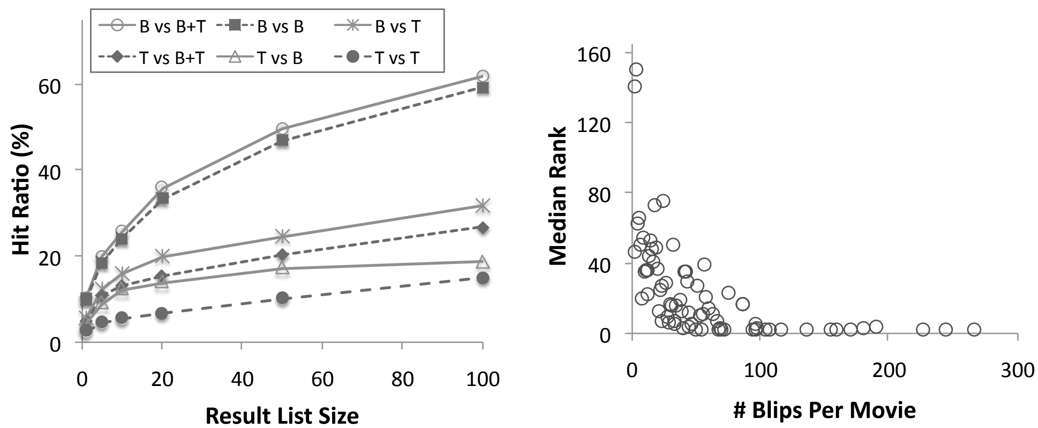

Figure 5: Hit ratios for the 6 query-index combinations (left) and median rank vs. movie index size (right)

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.

	movies	*apps*	*books*	*games*
# Products	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

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 $k1$ 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.

	movies	*apps*	*books*	*games*
$k1$	1	2	0.5	0
b	0.25	0.25	0.5	–

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-

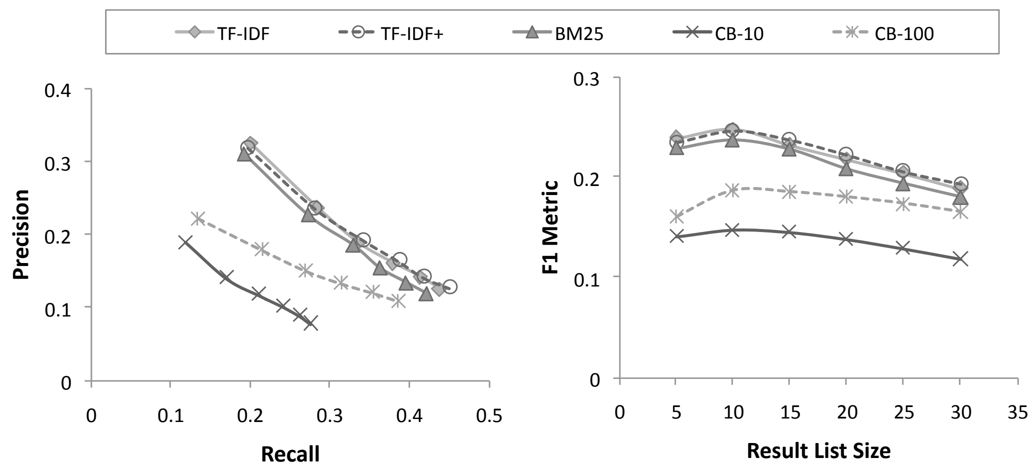

Figure 6: *Movies* dataset: precision-recall (left) and F1 metric (right) for user-based (TF-IDF vs. TF-IDF+ vs. BM25) and community-based (*CB*-10 vs. *CB*-100) recommendation.

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-

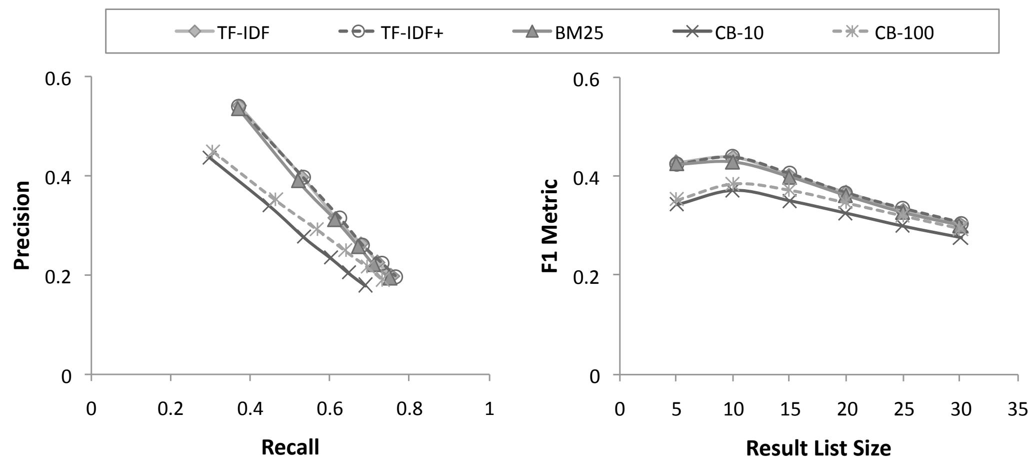

Figure 7: *Applications* dataset: precision-recall (left) and F1 metric (right) for user-based (TF-IDF vs. TF-IDF+ vs. BM25) and community-based (CB-10 vs. CB-100) recommendation.

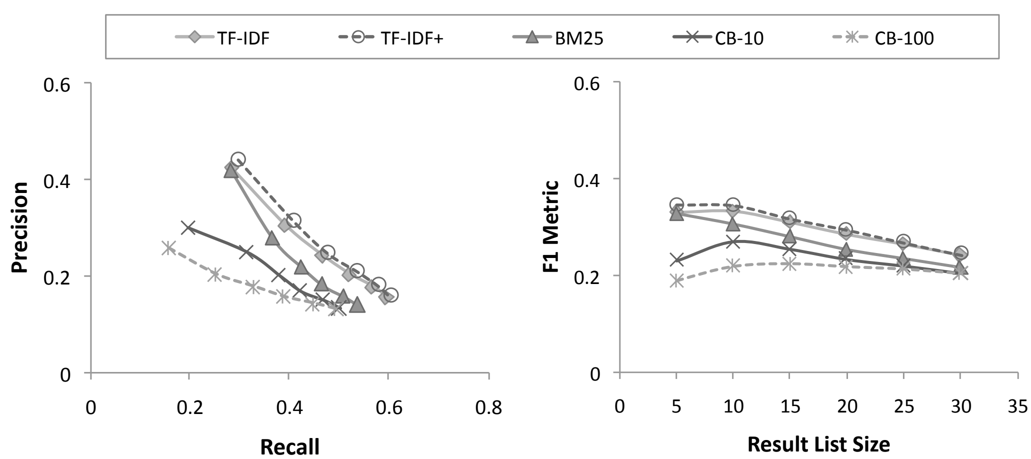

Figure 8: *Books* dataset: precision-recall (left) and F1 metric (right) for user-based (TF-IDF vs. TF-IDF+ vs. BM25) and community-based (CB-10 vs. CB-100) recommendation.

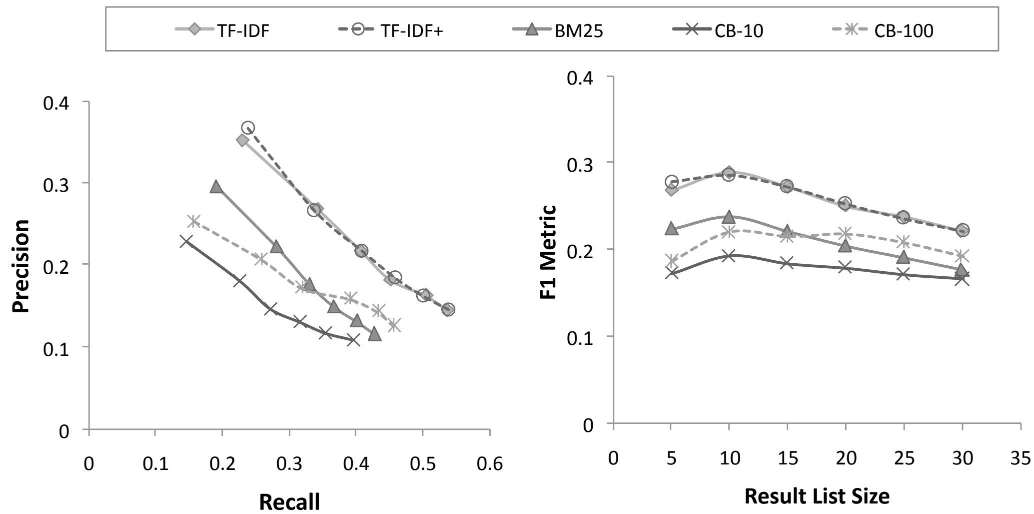

Figure 9: *Games* dataset: precision-recall (left) and F1 metric (right) for user-based (TF-IDF vs. TF-IDF+ vs. BM25) and community-based (*CB*-10 vs. *CB*-100) recommendation.

tain 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

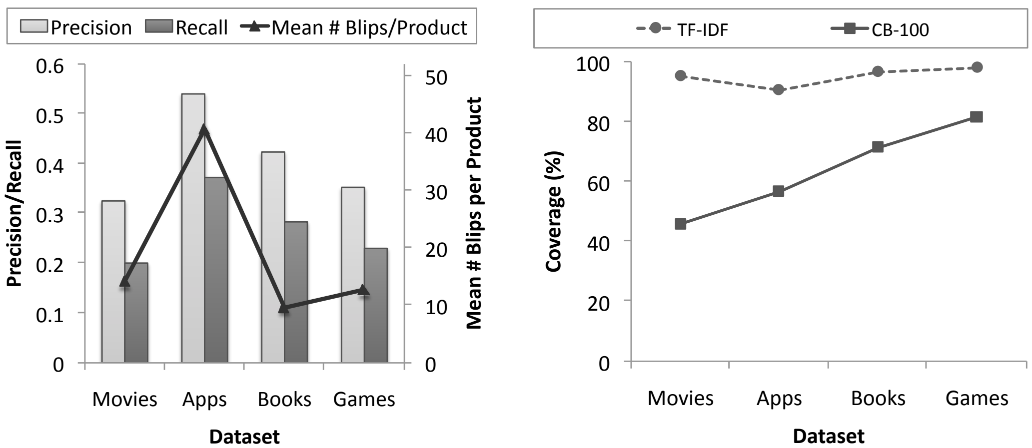

Figure 10: Precision and recall (for recommendation lists of size 5) provided by user-based recommendation using TF-IDF and mean number of blips per product for each dataset (left) and the coverage provided by the recommendation strategies for each dataset (right).

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

[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 (WI-IATW '06)*, 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] A. Angel, N. Koudas, N. Sarkas, and D. Srivastava. What's on the grapevine? In *Proceedings of the 35th SIGMOD international conference on Management of data (SIGMOD '09)*, pages 1047–1050, New York, NY, USA, 2009. ACM.

[5] S. Baccianella, A. Esuli, and F. Sebastiani. Multi-facet rating of product reviews. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval (ECIR '09)*, pages 461–472, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[7] N. Banerjee, D. Chakraborty, K. Dasgupta, S. Mittal, A. Joshi, S. Nagar, A. Rai, and S. Madan. User interests in social media sites: an exploration with micro-blogs. In *Proceeding of the 18th ACM conference on Information and knowledge management (CIKM '09)*, pages 1823–1826, New York, NY, USA, 2009. ACM.

[8] A. Bermingham and A. F. Smeaton. Classifying sentiment in microblogs: is brevity an advantage? In *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM '10)*, pages 1833–1836, New York, NY, USA, 2010. ACM.

[9] J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. F. Cooper and S. Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI '98)*, pages 43–52. Morgan Kaufmann, 1998.

[10] A. Brew, D. Greene, and P. Cunningham. Using crowdsourcing and active learning to track sentiment in online media. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI '10)*, pages 145–150, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[11] D. Bridge and P. Healy. Ghostwriter-2.0: Product reviews with case-based support. In M. Bramer, M. Petridis, and A. Hopgood, editors, *Proceedings of the Thirtieth International Conference on Innovative Techniques and Applications of Artificial Intelligence (SGAI '10)*, pages 467–480. Springer, 2010.

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

[13] I. Cantador, A. Bellogín, and D. Vallet. Content-based recommendation in social tagging systems. In *Proceedings of the fourth ACM conference on Recommender systems (RecSys '10)*, pages 237–240, New York, NY, USA, 2010. ACM.

[14] O. Celma and P. Herrera. A new approach to evaluating novel recommendations. In *Proceedings of the 2008 ACM conference on Recommender systems (RecSys '08)*, pages 179–186, New York, NY, USA, 2008. ACM.

[15] 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.

[16] F. L. Cruz, J. A. Troyano, F. Enríquez, F. J. Ortega, and C. G. Vallejo. A knowledge-rich approach to feature-based opinion extraction from product reviews. In *Proceedings of the 2nd international workshop on Search and mining user-generated contents (SMUC '10)*, pages 13–20, New York, NY, USA, 2010. ACM.

[17] L. Dey and S. K. M. Haque. Opinion mining from noisy text data. In *Proceedings of the second workshop on Analytics for noisy unstructured text data (AND '08)*, pages 83–90, New York, NY, USA, 2008. ACM.

[18] P. du Boucher-Ryan and D. Bridge. Collaborative recommending using formal concept analysis. *Knowledge Based Systems*, 19(5):309–315, 2006.

[19] C. Grier, K. Thomas, V. Paxson, and M. Zhang. @spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security (CCS '10)*, pages 27–37, New York, NY, USA, 2010. ACM.

[20] C. Hayes and P. Cunningham. Context boosting collaborative recommendations. *Knowledge Based Systems*, 2–4:131–138, 2004.

[21] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '99)*, pages 230–237, New York, NY, USA, 1999. ACM.

[22] 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.

[23] N. Jakob, S. H. Weber, M. C. Müller, and I. Gurevych. Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. In *Proceeding of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion (TSA '09)*, pages 57–64, New York, NY, USA, 2009. ACM.

[24] B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Micro-blogging as online word of mouth branding. In *Proceedings of the 27th international*

conference extended abstracts on Human factors in computing systems (CHI EA '09), pages 3859–3864, New York, New York, USA, 2009. ACM Press.

[25] N. Jindal and B. Liu. Opinion spam and analysis. In *Proceedings of the international conference on Web search and web data mining (WSDM '08)*, pages 219–230, New York, NY, USA, 2008. ACM.

[26] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD '09)*, pages 447–456, Paris, France, June 28–July 1 2009.

[27] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[28] 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 of the European conference on artificial intelligence (ECAI '06) workshop on recommender systems*, pages 62–66, Riva del Garda, Italy, 2006.

[29] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries (DL '00)*, pages 195–204, New York, NY, USA, 2000. ACM.

[30] 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 (EMNLP '04)*, pages 412–418, 2004.

[31] V. Pandey and C. Iyer. Sentiment analysis of microblogs, 2009. Accessed November 2010.

[32] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the 2002 conference on Empirical methods in natural language processing (EMNLP '02)*, pages 79–86, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[33] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The adaptive web: Methods and strategies of Web personalization*, pages 325–341. Springer-Verlag, Berlin, Heidelberg, 2007.

[34] J. Pérez-Iglesias, J. R. Pérez-Agüera, V. Fresno, and Y. Z. Feinstein. Integrating the Probabilistic Models BM25/BM25F into Lucene. *CoRR*, abs/0911.5046, 2009.

[35] O. Phelan, K. McCarthy, M. Bennett, and B. Smyth. Terms of a feather: content-based news recommendation and discovery using twitter. In *Proceedings of the 33rd European conference on Advances in information retrieval (ECIR '11)*, 2011.

[36] 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.

[37] A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT '05)*, pages 339–346, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[38] J. Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL Student Research Workshop (ACL '05)*, pages 43–48, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[39] 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.

[40] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Text REtrieval Conference (TREC)*, pages 109–126, 1996.

[41] S. E. Robertson and K. S. Jones. Simple, proven approaches to text retrieval. Technical report, Microsoft Research Ltd, University of Cambridge, Cambridge, UK, 1997.

[42] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[43] 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.

[44] 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.

[45] J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1-2):115–153, 2001.

[46] 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. ACM Press/Addison-Wesley Publishing Co., 1995.

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

[48] H. Tang, S. Tan, and X. Cheng. A survey on sentiment detection of reviews. *Expert Systems with Applications*, 36(7):10760–10773, 2009.

[49] C. J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 1979.

[50] 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.

[51] R. T. A. Wietsma and F. Ricci. Product reviews in mobile decision aid systems. In *Pervasive Mobile Interaction Devices (PERMID 2005)*, pages 15–18, Munich, Germany, 2005.

[52] M. Zhang and N. Hurley. Statistical modeling of diversity in top-n recommender systems. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT '09)*, volume 01, pages 490–497, Washington, DC, USA, 2009. IEEE Computer Society.

[53] 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.